**密 级：内部公开**

**版 本 号：V1.0**

**发布日期：2019年6月5日**

**实施日期：2019年月日**

**关系数据库编码规范**



**山东创德软件技术有限公司**

文件更改摘要：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **日期** | **版本号** | **修订说明** | **修订人** | **审核人** | **批准人** |
| 2019-06-05 | V1.0 | 新增 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**––––––––––––––––––––––––––––––––––––**

**山东创德软件技术有限公司对本文件资料享受著作权及其它专属权利，未经书面许可，不得将该等文件资料（其全部或任何部分）披露予任何第三方，或进行修改后使用。**

**目 录**

**[1. 概述 5](#_Toc330460611)**

**[1.1 简介 5](#_Toc330460612)**

**[1.2 约定 5](#_Toc330460613)**

**[1.3 范围 6](#_Toc330460614)**

**[1..4 参考 6](#_Toc330460615)**

**[2.书写规范 6](#_Toc330460616)**

**[2.1 大小写风格 6](#_Toc330460617)**

**[规则1 6](#_Toc330460618)**

**[2.2 缩进风格 6](#_Toc330460619)**

**[规则1 6](#_Toc330460620)**

**[规则2 6](#_Toc330460621)**

**[规则3 7](#_Toc330460622)**

**[建议1 7](#_Toc330460623)**

**[建议2 8](#_Toc330460624)**

**[2.3 空格及换行 9](#_Toc330460625)**

**[规则1 9](#_Toc330460626)**

**[规则2 9](#_Toc330460627)**

**[规则3 10](#_Toc330460628)**

**[规则4 11](#_Toc330460629)**

**[规测5 11](#_Toc330460630)**

**[2.4 其他 12](#_Toc330460631)**

**[规则1 12](#_Toc330460632)**

**[规则2 12](#_Toc330460633)**

**[规则3 13](#_Toc330460634)**

**[规则4 13](#_Toc330460635)**

**[3. 命名规范 13](#_Toc330460636)**

**[规则1 13](#_Toc330460637)**

**[规则2 15](#_Toc330460638)**

**[规则3 15](#_Toc330460639)**

**[规则4 15](#_Toc330460640)**

**[规则5 16](#_Toc330460641)**

**[规则6 16](#_Toc330460642)**

**[建议1 16](#_Toc330460643)**

**[4. 注释规范 17](#_Toc330460644)**

**[规则1 17](#_Toc330460645)**

**[规则2 17](#_Toc330460646)**

**[规则3 19](#_Toc330460647)**

**[规则4 19](#_Toc330460648)**

**[规则5 19](#_Toc330460649)**

**[规则6 19](#_Toc330460650)**

**[规则7 20](#_Toc330460651)**

**[规则9 20](#_Toc330460652)**

**[建议1 20](#_Toc330460653)**

**[建议2 20](#_Toc330460654)**

**[建议3 20](#_Toc330460655)**

**[建议4 21](#_Toc330460656)**

**[建议5 21](#_Toc330460657)**

**[5. 语法规范 22](#_Toc330460658)**

**[5.1 基本规则 22](#_Toc330460659)**

**[规则1 22](#_Toc330460660)**

**[规则2 22](#_Toc330460661)**

**[规则3 22](#_Toc330460662)**

**[规则4 22](#_Toc330460663)**

**[规则5 23](#_Toc330460664)**

**[5.2 性能规则 23](#_Toc330460665)**

**[规则1 23](#_Toc330460666)**

**[规则2 23](#_Toc330460667)**

**[规则3 24](#_Toc330460668)**

**[规则4 24](#_Toc330460669)**

**[建议1 24](#_Toc330460670)**

**[建议2 25](#_Toc330460671)**

**[建议3 25](#_Toc330460672)**

**[建议4 25](#_Toc330460673)**

**[6. 存储过程规范 26](#_Toc330460674)**

**[规则1 26](#_Toc330460675)**

**[规则2 26](#_Toc330460676)**

**[规则3 26](#_Toc330460677)**

**[规则4 27](#_Toc330460678)**

**[规则5 27](#_Toc330460679)**

**[规则6 27](#_Toc330460680)**

**[规则7 28](#_Toc330460681)**

**[建议1 28](#_Toc330460682)**

**[建议2 28](#_Toc330460683)**

**[7. 事务处理 29](#_Toc330460684)**

**[规则1 29](#_Toc330460685)**

**[规则2 29](#_Toc330460686)**

**[规则3 29](#_Toc330460687)**

**[规则4 29](#_Toc330460688)**

**[建议1 29](#_Toc330460689)**

**[建议2 30](#_Toc330460690)**

**[8. 设计规范 30](#_Toc330460691)**

**[8.1 数据库设计优化 30](#_Toc330460692)**

**[建议1 30](#_Toc330460693)**

**[建议2 30](#_Toc330460694)**

**[8.2 表设计 30](#_Toc330460695)**

**[规则1 30](#_Toc330460696)**

**[规则2 30](#_Toc330460697)**

**[建议1 31](#_Toc330460698)**

**[8.3 视图设计 31](#_Toc330460699)**

**[规则1 31](#_Toc330460700)**

**[规则2 31](#_Toc330460701)**

**[建议1 31](#_Toc330460702)**

**[建议2 31](#_Toc330460703)**

**[8.4 索引设计 32](#_Toc330460704)**

**[规则1 32](#_Toc330460705)**

**[规则2 32](#_Toc330460706)**

**[规则3 32](#_Toc330460707)**

**[建议1 32](#_Toc330460708)**

**[建议2 32](#_Toc330460709)**

**[建议3 32](#_Toc330460710)**

**[9. 附录 33](#_Toc330460711)**

**[9.1 保留关键字 33](#_Toc330460712)**

**[9.1.1 ODBC（SQL-92） 33](#_Toc330460713)**

**[9.1.2. SQL Server 35](#_Toc330460714)**

**[9.1.3. 未来版本的关键字 37](#_Toc330460715)**

**[9.2. 数据库设计范式 39](#_Toc330460716)**

**[9.2.1. 简介 39](#_Toc330460717)**

**[9.2.2. 第一范式（1NF）无重复的列 39](#_Toc330460718)**

**[9.2.3. 第二范式（2NF）属性 39](#_Toc330460719)**

**[9.2.4. 第三范式（3NF）属性 40](#_Toc330460720)**

# 概述

## 1.1 简介

本规范强调实用性、可操作性，根据开发人员在编码过程中常见的问题和易犯的错误，对代码编写的各方面均进行了详细明确的规范和约束。主要包括下列内容：

1. 书写规范
2. 命名规范
3. 注释规范
4. 语法规范
5. 存储过程规范
6. 事物处理
7. 设计规范

## 1.2 约定

本规范采用以下术语描述：

1. 规则：编程时强制必须遵守的原则。
2. 建议：编程时必须加以考虑的原则。
3. 说明：对此规则或建议进行必要的解释。
4. 示例：对此规则或建议从正、反两个方面的说明。

## 1.3 范围

本规范适用于鲁能软件使用主流关系数据库的编码等相关开发工作。

## 1..4 参考

1. 能源信息技术事业部开发资料之4\_关系数据库开发规范v1.02
2. 华为技术有限公司——MS SQL Server数据库编程规范

# 2.书写规范

## 2.1 大小写风格

### 规则1

所有数据库代码统一使用小写字母书写，有利于减少硬分析提高效率，也方便不同数据库的移植。

## 2.2 缩进风格

### 规则1

程序块采用缩进风格书写，保证代码清晰易读，风格一致。缩进格数统一为 4 格。

### 规则2

必须使用空格键，不允许使用TAB键。

### 规则3

注释必须保证左对齐。

### 建议1

insert...select语句中，每行保证在110列以内，并应使每行的字段顺序对应，以增强可读性。

示例：

insert into tablename

(field1, field2, field3, field4,

field5, field6,

field7)

select

field1, field2,

field3, field4, field5, field6,

field7

from tablename

应写成：

insert into tablename

(field1, field2, field3,

field4, field5, field6,

field7

)

select field1, field2, field3,

field4, field5, field6,

field7

from tablename

说明：

1. select语句中每行的字段应与insert语句对应。
2. insert语句中折行后的字段名应缩进并与上一行的第一个字段名对齐。
3. select语句中折行后的字段名应缩进并与上一行的第一个字段名对齐。

### 建议2

insert语句中，如果需要对每个字段增加注释，则应将每个字段单独列为一行，并在行尾增加注释。

示例：

insert into t\_desttablename

( destfieldname1, -- 注释1

destfieldname2 -- 注释2

)

values

( @fieldvalue1,

@fieldvalue2

)

## 2.3 空格及换行

### 规则1

不允许把多个短语句写在一行中，即一行只写一条语句。

示例：

select @variable1 = 1 select @variable3 = @variable1 + @variable2

应写成：

select @variable1 = 1

select @variable3 = @variable1 + @variable2

### 规则2

相对独立的程序块之间、变量说明之后必须加空行。

示例：

select @variable1 = 1

if @begindate is null

begin

select @beginDate = dateadd(day, 15, getdate())

...

end

应写成：

-- 初始化局部变量

select @variable1 = 1

-- 判断开始时间

if @begindate is null

begin

select @beginDate = dateadd(day, 15, getdate())

...

end

说明：两个程序块在逻辑上相对独立，应用空行加以分隔，同时增加注释。

### 规则3

超过110列的语句要分行书写，长表达式应在低优先级操作符处换行，操作符或关键字放在新行之首。划分出的新行应适当地缩进，使排版整齐，语句可读。

示例：

110列

↓

…… (a \* b \* c \* d) + (e \* f) + ……

应写成：

…… (a \* b \* c \* d)

+ (e \* f) + ……

说明：

1. 加法的优先级低于乘法，因此应在加号处折行。
2. 两组乘法虽然在逻辑上会先于加法执行，但显式加上括号使可读性更强。

### 规则4

双目运算符前后、操作符之后、间隔符、左括号之后以及右括号之前应以空格分隔。

示例：

select @dateStr=convert(varchar,getdate(),112)

select @var1=@var2+@var3

应写成：

select @datestr = convert( varchar, getdate(), 112 )

select @var1 = @var2 + @var3

### 规测5

定义两个以上变量或者给变量赋值时，分多行写，定义每一行对一个变量，便于注释。

示例：

declare @variable1 varchar(20),@variable2 varchar(100),@variable2 integer

应写成：

declare @variable1 varchar(20) --注释

declare @variable2 varchar(100) --注释

declare @variable3 varchar(200) --注释

## 2.4 其他

### 规则1

不允许写select \* from……，必须指明需要读取的具体字段。

### 规则2

if 语句后面的条件必须使用括号括起来。if语句中用begin...end结构，阅读起来方便些。

示例:

--如果可上传超出最大数

if @i\_sptonecount>= @i\_sptonemaxnum

return 3

应写成:

--如果可上传超出最大数

if ( @i\_sptonecount>= @i\_sptonemaxnum )

begin

return 3

end

### 规则3

表别名要简短。通常，使用小写的字母作为别名。

示例:

select a.phonenumber as userphone, b.tonename as userdowntone

from t\_personallib a, t\_webtonelib b

where a.tonecode = b.tonecode

and a.phonenumber = @phonenumber

### 规则4

当一个 T-SQL 语句中涉及到多个表时，始终使用表名别名来限定字段名。这使其他人阅读起来更清楚，避免了含义模糊的引用。

示例:

select phonenumber, b.tonename as userdowntone

from t\_personallib a, t\_webtonelib b

应写成：

select a.phonenumber as userphone, b.tonename as userdowntone

from t\_personallib a, t\_webtonelib b

# 3. 命名规范

### 规则1

所有用户自定义的数据库对象名称统一使用形如“小写首字母+下划线+混合拼写”的格式，其中小写首字母表示对象类型。具体规则参见下表。

|  |  |  |  |
| --- | --- | --- | --- |
| 对象类型 | 首字母 | 范例 | 备注 |
| 表(Table) | t | t\_tablename |  |
| 视图(View) | v | v\_viewname |  |
| 索引(Index) | ix,icx | ix\_tablename\_fieldname | 如果表名或字段名过长，则用表名和字段名的缩写表示  icx表示聚集索引 |
| 主键(Primary Key) | pk | pk\_tablename | 如果表名过长，则用表名的缩写表示 |
| 外键(Foreign Key) | fk | fk\_tablename1\_tablename2\_fieldname | tablename1为外键所在的表，tablename1为外键所参照的表 |
| 触发器(Trigger) | tr | tr\_talbename\_triggername |  |
| 过程(Stored Procedure) | p | p\_admin\_procedurename  p\_public\_procedurename | p\_模块名称\_功能描述  功能模块目前有:ivr,admin,sp,user,cs,corp对于各模块共用的使用: public |
| 函数(Function) | f | f\_user\_getdate  f\_public\_uploadtone | f\_模块名称\_功能描述  功能模块目前有ivr,admin,sp,user,cs,corp对于各模块共用的使用 public |
| 自定义类型 | t\_type | t\_type\_phonenumber | t\_type\_字段名 |

注意：

1. 禁止与所使用的数据库中系统表的前缀相同。例如Sql Server系统对象多以sys开头，因此编码时禁止以sys\_object\_name形式为用户创建的对象命名。

### 规则2

所有标示符、对象名的长度不得超过30个字符。

说明：

因为我们不同的数据库对外提供的是统一接口，由于Oracle在标示符命名上有30个字符的限制，所以在所有命名中长度不得超过30个字符。

### 规则3

标识符的命名使用有意义、易于记忆、描述性强、简短及具有唯一性的英文单词。要清晰、明了，有明确含义，使用完整的英文单词或大家基本可以理解的缩写，避免使人产生误解。不允许使用除小写字母、数字、\_、@以外的其他字符。

说明：

个人的命名风格，在符合所在项目组或产品组的命名规则的前提下，才可使用。（即命名规则中没有规定到的地方才可有个人命名风格）。较短的单词可通过去掉“元音”形成缩写；较长的单词可取单词的头几个字母形成缩写；一些单词有大家公认的缩写。

### 规则4

命名中若使用特殊约定或缩写，则要有注释说明。

### 规则5

对于变量命名，禁止取单个字符（如i、j、k...），建议除了要有具体含义外，还能表明其变量类型、数据类型等。形如i、j、k的变量只允许作为局部循环变量。

说明：

变量，尤其是局部变量，如果用单个字符表示，很容易敲错（如i写成j），而编译时又检查不出来，有可能为了这个小小的错误而花费大量的查错时间。

### 规则6

所有的标示符、对象名命名不得使用保留字、关键字。

说明：保留关键字参见[附录](#_保留关键字)。

### 建议1

用正确的反义词组命名具有互斥意义的变量或相反动作的函数等。

示例：

begin - end 开始 - 结束

start - finish 启动 - 完成

first - last 第一个 - 最后一个

prior - next 前一个 - 后一个

add - remove 增加 - 删除

create - destroy 创建 - 析构

get - set 获取 - 设置

increment - decrement 增量 - 减量

lock - unLock加锁 - 解锁

open - close 打开 - 关闭

# 4. 注释规范

### 规则1

一般情况下，源程序有效注释量必须在30％以上。

说明：

注释的原则是有助于对程序的阅读理解，在该加的地方都加了，注释不宜太多也不能太少，注释语言必须准确、易懂、简洁。

### 规则2

在数据库脚本文件头部应进行注释，注释必须列出：版权说明、版本号、生成日期、作者、内容、功能、修改日志等，头文件的注释中还应有函数或过程功能简要说明。

示例:使用以下书写规范

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: f\_back\_createbaktable

PURPOSE: 通过复制原表结构创建日志表

REVISIONS:

Ver Date Author Description

----------------------------------------------------------------------------

D20 2005-07-15 authorname1.创建此存储过程

---------------------------------------------------------------------------

D20SP2 2005-09-05 authorname1.调整程序结构

2.增加索引的复制

---------------------------------------------------------------------------

RETURN:

0 成功

1 失败

NOTES:

1.本函数只支持唯一索引和普通B-TREE索引的复制，不支持特殊索引;

2.支持主键，但不支持外键复制;

3.支持非空约束，但不支持其他特殊约束。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

说明：

NAME：函数名称；

PURPOST：函数的用途；

REVISIONS下面是版本信息：

Ver：当前版本

Date：创建或修改日期

Author：创建人或修改人的姓名

Description：在修改函数时一定要在这里写出改动的内容，用1、2、3清晰的列出来。

NOTES：使用该函数特别需要注意的事情，如果没有可以不写。

### 规则3

边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除。

### 规则4

注释的内容要清楚、明了，含义准确，防止注释二义性，在注释中不允许使用缩写。

说明：

错误的注释不但无益反而有害。

### 规则5

注释应与其描述的代码相近，对代码的注释应放在其上方或右方（对单条语句的注释）相邻位置，不可放在下面，如放于上方则需与其上面的代码用空行隔开。不允许在一行代码或表达式的中间插入注释。

### 规则6

对变量的定义和分支语句（条件分支、循环语句等）必须编写注释。

说明：

这些语句往往是程序实现某一特定功能的关键，对于维护人员来说，良好的注释帮助更好的理解程序，有时甚至优于看设计文档。

### 规则7

将注释与其上面的代码用空行隔开。

### 规则9

语句、参数、变量后的注释采用左对齐方式。

### 建议1

在程序块的结束行右方加注释标记，以表明某程序块的结束。

说明：当代码段较长，特别是多重嵌套时，这样做可以使代码更清晰，更便于阅读。

### 建议2

对于单行注释使用“--”，对于多行注释可考虑使用“/\*\*/”。

### 建议3

通过对存储过程、变量、结构等正确的命名以及合理地组织代码的结构，使代码成为自注释的。

说明：

清晰准确的函数、变量等的命名，可增加代码可读性，并减少不必要的注释。

### 建议4

在代码的功能、意图层次上进行注释，提供有用、额外的信息。

说明：

注释的目的是解释代码的目的、功能和采用的方法，提供代码以外的信息，帮助读者理解代码，防止没必要的重复注释信息。

示例：

如下注释意义不大。

-- @retcode = 0

if @retcode = 0

而如下的注释则给出了额外有用的信息。

-- 流程传入的标志为发送成功

if ( @retcode = 0 )

### 建议5

注释应考虑程序易读及外观排版的因素，使用的语言若是中、英兼有的，建议多使用中文，除非能用非常流利准确的英文表达。

说明：

注释语言不统一，影响程序易读性和外观排版，出于对维护人员的考虑，建议使用中文。

# 5. 语法规范

## 5.1 基本规则

### 规则1

避免在业务代码使用系统存储过程，如xp\_cmdshell。

### 规则2

判断变量是否为空，应使用 is null 或 is not null 进行比较。对于如果可能为空的变量使用，需要先判断是否为空，或者使用isnull函数进行空值的转换。

### 规则3

确保变量和参数的类型和大小与表数据列相匹配。

### 规则4

select、insert 子句的代码中不允许出现“\*”以代替查询的所有字段，必须用实际的字段名代替（count(\*) 除外）。

示例：

错误用法：

insert into t\_tablename values (a,b)

正确用法：

insert into t\_tablename

( field1,

field2

)

values

（ ‘a’,

‘b’

) .

### 规则5

储存过程或者函数的in 、out参数应按类别分开书写，不要交叉。

## 5.2 性能规则

### 规则1

避免对where子句中的字段使用函数。禁止对where子句中的字段进行计算。

### 规则2

使用where子句替换having子句，having只对统计结果进行过滤。

示例：

select tonecode count(tonecode)

from t\_personallib

group by tonecode

having tonecode> ‘600666000000’

应写成：

select tonecode count(tonecode)

from t\_personallib

where tonecode> ‘600666000000’

group by tonecode

### 规则3

在长度允许范围内，使用integer代替numeric

示例：

declare @var numeric(10,2)

select @var = 10

select \* from t\_table1 where Field2 = @var

应写成：

declare @var integer

select @var = 10

select \* from t\_table1 where Field2 = @var

### 规则4

禁止在储存过程或者函数中反复访问同一张表，尤其数据量较大的表，可以考虑根据条件先将数据提取到临时表或者临时表变量中，再对临时表进行操作。

### 建议1

对于比较复杂的语句，或者涉及数据量较大的表的语句，可以通过做查询计划检测、调整语句性能。

### 建议2

用case语句合并多重表扫描。

我们常常必须基于多组数据表计算不同的聚集。例如下例通过三个独立查询：

1）select count(1) from emp where sal<1000;

     2）select count(1) from emp where sal between 1000 and 5000;

     3）select count(1) from emp where sal>5000;

这样我们需要进行三次全表查询，但是如果我们使用case语句：

    select count(case when sal<1000 then 1 else null end ),

count(case when sal between 1000 and 5000 then 1 else null end),

count (case when sal>5000 then 1 else null end)

from emp

这样查询的结果一样，但是执行计划只进行了一次全表查询。

### 建议3

like子句尽量前端匹配。

因为like参数使用的非常频繁，因此如果能够对like子句使用索引，将很高的提高查询的效率。

### 建议4

尽量不使用 "<>"。

尽量去掉 "<>"，避免全表扫描，如果数据是枚举值，且取值范围固定，则修改为"OR"方式。

update serviceinfo set state=0 where state<>0;

以上语句由于其中包含了"<>"，执行计划中用了全表扫描,没有用到state字段上的索引。实际应用中，由于业务逻辑的限制，字段state为枚举值，只能等于0，1或2，而且，值等于=1，2的很少，因此可以去掉"<>"，利用索引来提高效率。

修改为：

update serviceinfo set state=0  where state = 1 or state = 2 。

# 6. 存储过程规范

### 规则1

代码中声明与表的字段相对应的变量时，应保证变量名和字段名相同，数据类型与数据长度相同；对已有定义自定义类型，必须使用自定义类型, 增强可读性。

### 规则2

存储过程中变量的声明应集中在AS和BEGIN关键字之间，不允许在代码中随处定义变量。

### 规则3

始终指定十进制数据类型的精度和范围。

### 规则4

如果存储过程中都会用到getdate()置值,那么就在程序开始时定义一个赋值为getdate()的供整个存储过程使用的变量。

### 规则5

存储过程中每个失败出口均应返回不同的自定义错误码，提高代码的可调试性。

### 规则6

存储过程的代码必须放在begin...end块中。

示例：

create procedure p\_procedure (

...

)

as

declare ...

...

return

go

应写成：

create procedure p\_procedure (

...

)

as

declare ...

begin

...

return

end

go

说明：这样要求的目的是提高代码的可读性，而且便于代码的移植。

### 规则7

每个存储过程内的代码前必须加上SET NOCOUNT ON ，结尾必须加上SET NOCOUNT OFF，SET NOCOUNT ON使存储过程不返回受影响的行数。

### 建议1

Insert、update、delete操作的存储过程中都有@intEffectedCntint输出参数，表示本次操作造成数据库受影响的记录数。

### 建议2

确保使用到所有变量和参数。不建议存在没有使用到的变量、参数。

# 7. 事务处理

### 规则1

诸如删除/清空临时表之类对数据完整性要求不高的操作，禁止放在事务中处理。

### 规则2

为保证数据的完整性，在存在显式事务的函数中,必须加上SET XACT\_ABORT ON

说明：

加上SET XACT\_ABORT ON ,在事务内,如果不是语法错误,在执行出错后,事务将不再往下执行,整个事务将自动回滚。这样就无需在每条语句后加上 if @@error <> 0 的判断了。

### 规则3

尽量避免使用嵌套事务。

### 规则4

不允许在存储过程中编写DDL语言,如临时去掉一个表的约束等,建立临时表及表变量的语句除外。

### 建议1

在某些特殊情况下，为了提高代码效率，可以不使用事务控制，但必须通过增加其他的容错处理以保证业务逻辑的正确性。

### 建议2

尽量避免大事务操作，慎用holdlock子句，提高系统的并发能力。

# 8. 设计规范

## 8.1 数据库设计优化

### 建议1

若无特殊需求，在数据库中不要使用存储过程、函数等涉及到业务逻辑的定义。

### 建议2

不过多使用触发器。触发器具有较强的隐蔽性，对表结构调整时容易忽略，造成数据错误。因此建议不要过多使用触发器。

## 8.2 表设计

### 规则1

如果某个字段值有多个含义，需要在相应的字段说明里说明各个值的含义。包括设计文档和脚本。

### 规则2

每个表都要设置主键。如果无现成的字段可用作主键，建议添加一个Guid字符列。

### 建议1

尽量通过业务代码保证数据的一致性，不要过多依赖主外键来保持数据一致性, 主外键关系会降低更新及插入数据的性能。

## 8.3 视图设计

### 规则1

不要使用复杂视图（三个或三个以上表的连接）。

JOIN三张以上表的视图，表的连接需要耗费CPU资源，不宜使用视图,对于经常使用的视图，建议将视图查询的数据保存到一张表中，定时刷新该表的数据来代替视图(或查询)。

### 规则2

视图中不能再引用视图。

### 建议1

视图建立特定的数据子集可以供任何数据库用户使用。

### 建议2

通过视图可以隐藏基表。

可以禁止所有用户访问数据库表，而要求用户只能通过视图操作数据。这种方法可以保护用户和应用程序不受某些数据库修改的影响。例如，如果可以创建一个名为"本月销售额"的视图。在每月的第一天，可以相应地修改视图的定义。如果没有这个视图，用户每月都必须重新编写查询以选择相应月份的销售额。

## 8.4 索引设计

### 规则1

创建索引必须有明确的使用目的，不允许创建无用的索引。

索引的存储需要占用空间，更新索引也需要耗用资源，创建无用的索引会降低性能和浪费存储空间，创建索引的脚本应该注明创建索引的理由。

### 规则2

依照常用的查询条件和数据分布来决定索引的创建。

### 规则3

索引字段不宜过多。对所选的每一列，可指出索引是按升序还是降序组织列值。

### 建议1

不要在过长的字符型列上建立索引，如长度超过50的varchar字段。

### 建议2

在不同的设备上创建文件或文件组，并将索引和表分别建在不同的文件（组）上。

### 建议3

集聚索引应当建立在需要经常按照范围查询的字段上，保证索引顺序与物理位置一致。

# 9. 附录

## 9.1 保留关键字

### 9.1.1 ODBC（SQL-92）

|  |  |  |
| --- | --- | --- |
| absolute | exec | overlaps |
| action | execute | pad |
| ada | exists | partial |
| add | external | pascal |
| all | extract | position |
| allocate | false | precision |
| alter | fetch | prepare |
| and | first | preserve |
| any | float | primary |
| are | for | prior |
| as | foreign | privileges |
| asc | fortran | procedure |
| assertion | found | public |
| at | from | read |
| authorization | full | real |
| avg | get | references |
| begin | global | relative |
| between | go | restrict |
| bit | goto | revoke |
| bit\_length | grant | right |
| both | group | rollback |
| by | having | rows |
| cascade | hour | schema |
| cascaded | identity | scroll |
| case | immediate | second |
| cast | in | section |
| catalog | include | select |
| char | index | session |
| char\_length | indicator | session\_user |
| character | initially | set |
| character\_length | inner | size |
| check | input | smallint |
| close | insensitive | some |
| coalesce | insert | space |
| collate | int | sql |
| collation | integer | sqlca |
| column | intersect | sqlcode |
| commit | interval | sqlerror |
| connect | into | sqlstate |
| connection | is | sqlwarning |
| constraint | isolation | substring |
| constraints | join | sum |
| continue | key | system\_user |
| convert | language | table |
| corresponding | last | temporary |
| count | leading | then |
| create | left | time |
| cross | level | timestamp |
| current | like | timezone\_hour |
| current\_date | local | timezone\_minute |
| current\_time | lower | to |
| current\_timestamp | match | trailing |
| current\_user | max | transaction |
| cursor | min | translate |
| date | minute | translation |
| day | module | trim |
| deallocate | month | true |
| dec | names | union |
| decimal | national | unique |
| declare | natural | unknown |
| default | nchar | update |
| deferrable | next | upper |
| deferred | no | usage |
| delete | none | user |
| desc | not | using |
| describe | null | value |
| descriptor | nullif | values |
| diagnostics | numeric | varchar |
| disconnect | octet\_length | varying |
| distinct | of | view |
| domain | on | when |
| double | only | whenever |
| drop | open | where |
| else | option | with |
| end | or | work |
| end-exec | order | write |
| escape | outer | year |
| except | output | zone |
| exception |  |  |

### 9.1.2.SQL Server

Microsoft® SQLServer™ 2000

|  |  |  |
| --- | --- | --- |
| add | except | percent |
| all | exec | plan |
| alter | execute | precision |
| and | exists | primary |
| any | exit | print |
| as | fetch | proc |
| asc | file | procedure |
| authorization | fillfactor | public |
| backup | for | raiserror |
| begin | foreign | read |
| between | freetext | readtext |
| break | freetexttable | reconfigure |
| browse | from | references |
| bulk | full | replication |
| by | function | restore |
| cascade | goto | restrict |
| case | grant | return |
| check | group | revoke |
| checkpoint | having | right |
| close | holdlock | rollback |
| clustered | identity | rowcount |
| coalesce | identity\_insert | rowguidcol |
| collate | identitycol | rule |
| column | if | save |
| commit | in | schema |
| compute | index | select |
| constraint | inner | session\_user |
| contains | insert | set |
| containstable | intersect | setuser |
| continue | into | shutdown |
| convert | is | some |
| create | join | statistics |
| cross | key | system\_user |
| current | kill | table |
| current\_date | left | textsize |
| current\_time | like | then |
| current\_timestamp | lineno | to |
| current\_user | load | top |
| cursor | national | tran |
| database | nocheck | transaction |
| dbcc | nonclustered | trigger |
| deallocate | not | truncate |
| declare | null | tsequal |
| default | nullif | union |
| delete | of | unique |
| deny | off | update |
| desc | offsets | updatetext |
| disk | on | use |
| distinct | open | user |
| distributed | opendatasource | values |
| double | openquery | varying |
| drop | openrowset | view |
| dummy | openxml | waitfor |
| dump | option | when |
| else | or | where |
| end | order | while |
| errlvl | outer | with |
| escape | over | writetext |

### 9.1.3. 未来版本的关键字

|  |  |  |
| --- | --- | --- |
| absolute | found | preserve |
| action | free | prior |
| admin | general | privileges |
| after | get | reads |
| aggregate | global | real |
| alias | go | recursive |
| allocate | grouping | ref |
| are | host | referencing |
| array | hour | relative |
| assertion | ignore | result |
| at | immediate | returns |
| before | indicator | role |
| binary | initialize | rollup |
| bit | initially | routine |
| blob | inout | row |
| boolean | input | rows |
| both | int | savepoint |
| breadth | integer | scroll |
| call | interval | scope |
| cascaded | isolation | search |
| cast | iterate | second |
| catalog | language | section |
| char | large | sequence |
| character | last | session |
| class | lateral | sets |
| clob | leading | size |
| collation | less | smallint |
| completion | level | space |
| connect | limit | specific |
| connection | local | specifictype |
| constraints | localtime | sql |
| constructor | localtimestamp | sqlexception |
| corresponding | locator | sqlstate |
| cube | map | sqlwarning |
| current\_path | match | start |
| current\_role | minute | state |
| cycle | modifies | statement |
| data | modify | static |
| date | module | structure |
| day | month | temporary |
| dec | names | terminate |
| decimal | natural | than |
| deferrable | nchar | time |
| deferred | nclob | timestamp |
| depth | new | timezone\_hour |
| deref | next | timezone\_minute |
| describe | no | trailing |
| descriptor | none | translation |
| destroy | numeric | treat |
| destructor | object | true |
| deterministic | old | under |
| dictionary | only | unknown |
| diagnostics | operation | unnest |
| disconnect | ordinality | usage |
| domain | out | using |
| dynamic | output | value |
| each | pad | varchar |
| end-exec | parameter | variable |
| equals | parameters | whenever |
| every | partial | without |
| exception | path | work |
| external | postfix | write |
| flase | prefix | year |
| first | preorder | zone |
| float | prepare |  |

## 9.2. 数据库设计范式

### 9.2.1. 简介

关系数据库中的关系必须满足一定的要求，即满足不同的范式。

目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、第四范式（4NF）、第五范式（5NF）和第六范式（6NF）。满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多要求的称为第二范式（2NF），其余范式以次类推。一般说来，数据库只需满足第三范式（3NF）就行了。

### 9.2.2. 第一范式（1NF）无重复的列

所谓第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。在第一范式（1NF）中表的每一行只包含一个实例的信息。简而言之，第一范式就是无重复的列。

说明：在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

### 9.2.3. 第二范式（2NF）属性

完全依赖于主键[消除非主属性对主码的部分函数依赖]

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。例如员工信息表中加上了员工编号（emp\_id）列，因为每个员工的员工编号是唯一的，因此每个员工可以被唯一区分。这个唯一属性列被称为主关键字或主键、主码。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。简而言之，第二范式就是属性完全依赖于主键。

### 9.2.4. 第三范式（3NF）属性

不依赖于其它非主属性[消除传递依赖]

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（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。例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性。