应遵循的PL_SQL编码规则 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/268/2021_2022__E5_BA_94_E9_81_B5_E5_BE_AA_E7_c102_268285.htm 所有工作都独自完成 我们很少有人是孤立工作的；大多数PL/SQL开发工作是在相对较大的机构中进行的。但我们基本上还是在自己的小隔间里用自己的设备独自工作。几乎没有PL/SQL开发小组进行正规的代码复查或系统测试。 我不可能通过这篇文章改变你们开发小组的基本状态。因此，我仔细地选取出以下几点建议。实施其中任何一点并不需征得管理人员同意。不论你的小组是大是小，都不必让其中的每个人都赞同这些编码规则。你只需按以下建议来改变你的本人的编码方式： 1. 严格遵循命名约定，好像它们就是你的生命支柱。 2. 戒除编写SQL的嗜好：编写的SQL越少越好。 3. 使执行部分短小：告别"意大利面条式的代码"。 4. 找一位伙伴：非常赞同找个人来监督你的工作。

---

1. 遵循命名约定 如果你建立并严格遵循一套命名约定，特别是对于应用程序组件的，你就可以节省很多时间。 当然，遵循命名约定的想法并没有什么新意，你可能已经听烦了。所以我并不提出什么宏伟的命名计划，而是给出一些非常具体而明确的约定，然后证明这些约定会多么有用。 前几个月我一直在为PL/SQL开发人员设计、构建一种新工具。它名为Swyg（可以在www.swyg.com中找到），可以帮助程序员完成代码的生成、测试及重用的工作。它具有几个独特的组件。我为每个组件指定了一个由两个字母组

成的缩写名称，如下所示： SF-Swyg的基础部件 SM-Swyg的元数据 SG-Swyg的生成程序 SL-Swyg的代码库 ST-Swyg的单元测试 于是，我便遵循表1中的命名约定，同时使用这些缩写。 遵循这些约定有什么好处呢？一般来讲，如果我要求一致的命名规则，我就可以更流畅更高效地编写代码。 明确地说，这些约定具有可预测性，意思是说我编写的SQL程序能生成有用的脚本。例如，通过使用表1中的约定，可以生成Swyg中所有基础包的安装脚本。执行这些工作的SQL*Plus脚本如清单1所示。这类脚本非常有用，因为它意味着我不必手动维护安装脚本。当我向Swyg方案中增加另一个表，并生成一组相关包时，我只要运行我的脚本，更新后的安装脚本便会跳出来。

---

2. 戒除编写SQL的嗜好 编写的SQL越少越好，这似乎与我们的直觉不太一致。对于PL/SQL开发人员来说，这是一个奇特的建议，因为PL/SQL的主要优点之一就是可以毫不费力地在代码中编写SQL语句。不过，这种简易性也是这种语言的一个致命的弱点。 可以将纯粹的SQL语句直接置于PL/SQL代码中，而无需JDBC或ODBC之类的中间层。因此，无论何时何地，PL/SQL开发人员只要需要SQL语句，他们通常就会向其应用程序代码中嵌入SQL语句。那么这样做有什么问题吗？ 在PL/SQL代码中到处使用SQL语句必然会导致以下后果： 尽管实际表现不同，但同一逻辑语句仍会出现重复，从而导致过多的语法分析，且难于优化应用程序的性能。 暴露商务规则和方案。这直接在SQL语句中包含了执行商务规则的逻辑。这些规则总在变化，所以应用程序的

维护成本会急剧增加。 当然，你要编写的每一个PL/SQL应用程序几乎都是基于基础表和视图的。你需要执行SQL语句。问题不在于是否执行，而是何时执行、如何执行。 如果你对数据结构进行封装，或者将它们隐藏于一个PL/SQL代码层（通常是一个代码包）之后，那么你的应用程序将会更健壮，而且你还会发现创建和维护变得更易多了。