收获，不止 SQL 优化

第 十三 章

用工具进行 SQL 整体优化

E-Mail:45240040@qq.com

1 集合操作可以减少 PLSQL 与 SQL 的上下文切换开销，使得程序更加快速
2 集合操作避免某些函数或者内部的递归调用的频繁操作。
3 集合操作的思想正是 SQL 最原始的定义，即面向集合的。减少操作的重复次数，
使用一次操作可以尽可以处理比较多的数据。

动态优化案例分享：
背景：
某个业务功能模块上线后，ORACLE 9I 数据库里硬解析比较多，SHARE POOL 里 CALL PACKAGENAME.PROCEDURENAME(:1)这样语法的 SQL 下边挂着大量的 CHIND NUMBER。
经过与开发人员了解，业务逻辑里这样的，PACKAGE 里有多个 PROCEDURE，关键的两个 PROCEDURE 命名为 PROC1，PROC2 其中有个重要的表 A，该表中
有一列 SAP_PROC 是 VARCHAR2 类型的，放着 CALL PROC2(:1)，CALL PROC2(:1,:2...)之类，有一列是数值因子。
在 PROC1 这个过程中的逻辑是将 A 中的数据放到集合中去，根据 PROC1 调用其他过程计算出来一个因子，判断该因子是否与 A 中的因子是否对应上，
如果对应上，则执行 SAP_PROC 中对应的 SQL。使用的是 EXECUTE IMMEDIATE ....USING ...语法。这类即使是同样的文本，参数个数也一样的，都出来了硬解析。

分析过程:
一，认为在 JAVA 中有设置了 ORACLE 参数等环境变量导致 CHIND NUMBER 比较多,但检查了代码，没发现有。
二，认为绑定变量的值长度出现了变化导致的，但从开发了解，长度变化也就是 1 到 200 这样，而一个 SQL 硬解析出来的有几千个，
所以也排除了这种假设。
三，最后没想到其他的方面了，那么只能一步步去 DEBUG 程序，首先将 PROC2 里边的 SQL 内容都清空，也不行，此时也排除了 PROC2 本身的问题。
接下来想到会不会将 SQL 存入了表中，再取出来动态执行出现了问题？根据业务大体逻辑，构造了一些数据及环境，在测试上模拟，发现
也都有大量硬解析。此时虽然没找到根本原因,但知道是这种执行 SQL 方式上导致问题的了，而开发也尝试去修改代码，因为在 PROC1 中调用同个
PACKAGE 下的过程 PROC2，没必要用 CALL PACKAGENAME.PROCEDURENAME 这种方式，完全

直接就 PROC2 这样调用，只需要增加一些判断语句，
使得逻辑上与原因相同。
经过修改后，问题解决。

到此，问题解决，但根本原因未找到。

接下来需要做的，将该实现逻辑在 ORACLE 10G 11G 上测试，确认是否也有同样问题，确认是否是 9I 不支持，还是其他新版本也不支持，以便后续经验总结积累。