



Oracle Performance SQL Tuning

演讲人：崔光斌



课程介绍

- ❖ 本次培训内容主要是针对Oracle数据库SQL的性能优化
- ❖ 将会学习到的内容：
 1. Oracle架构与基础知识
 - ❖ 实例(Instance)、服务进程(Server processes)、SGA
 - ❖ 物理结构与逻辑结构
 2. 性能调优概述
 3. Oracle 表、索引
 4. Oracle 函数、存储过程

课程介绍(续1)

- ❖ 将会学习到的内容(续1):
 5. Oracle的SQL语句的处理过程与事务
 6. Oracle的Optimizer介绍(How to Explain plan)
 7. 学习Oracle的Join方式
 8. 基于Rule的Optimizer SQL性能分析
 9. 基于Cost的Optimizer SQL性能分析
 10. Oracle Hint使用方法与原则
 11. 使用SQL Trace 与 TKPROF工具



课程介绍(续2)

- ❖ 本培训对学员的要求：
 - ❧ 从事计算机Oracle数据库方面的开发人员
 - ❧ 从事计算机Oracle数据库的管理员
 - ❧ 已经对Oracle有基础的了解
 - ❧ 已经会写Oracle的SQL PL/SQL



课程介绍(续3)

- ❖ 本培训结束后学员能达到的能力目标：
 - ❧ 理解Oracle开发与性能的关系
 - ❧ 写出更好的SQL语句
 - ❧ 熟练找出数据库应用的程序的瓶颈并做优化
 - ❧ 具有中高级Oracle的DBA的水平



Oracle架构与基础知识

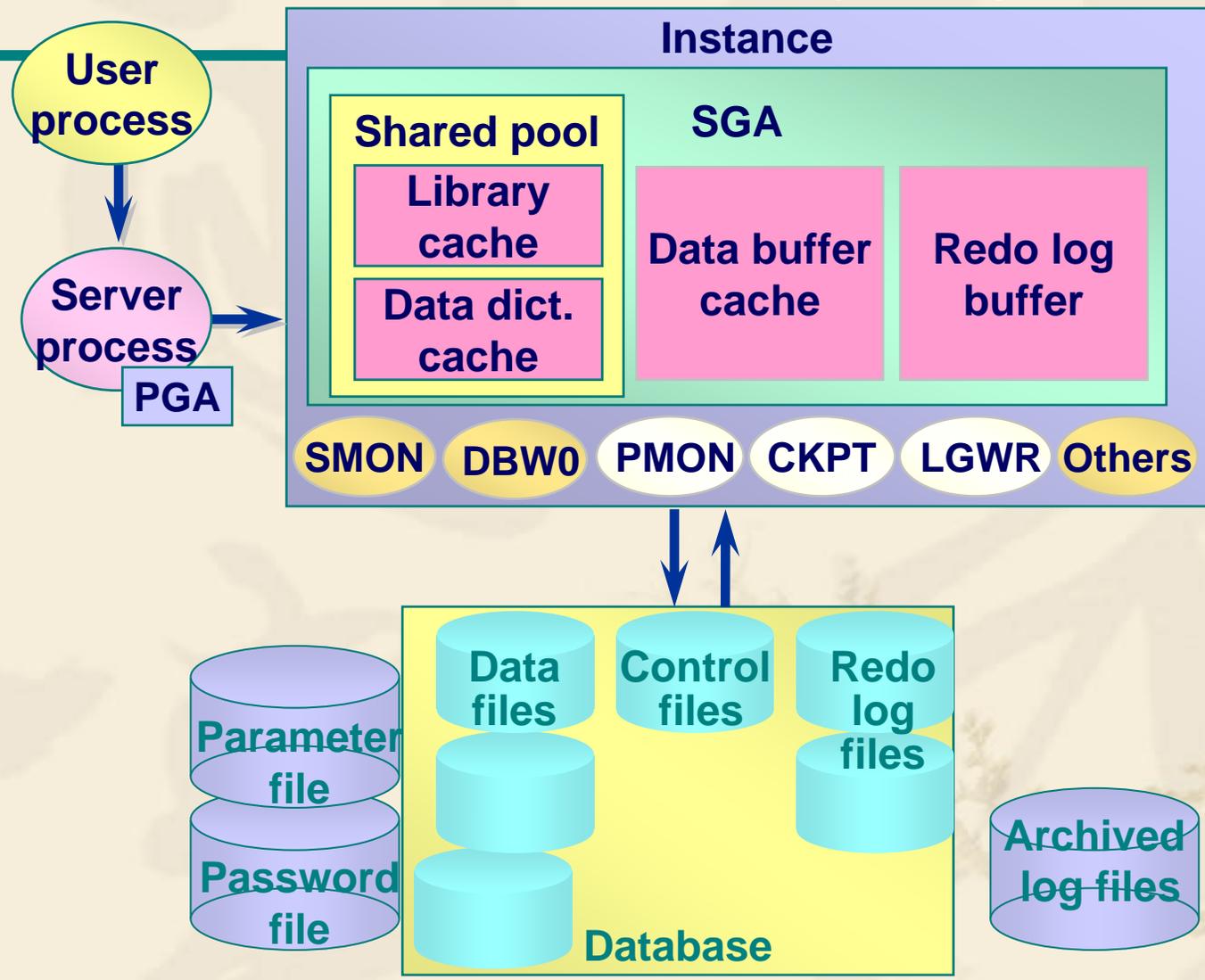
- ❖ 本课将要学习的内容：
 - ☞ Oracle的组件
 - ☞ 连接到Oracle Instance的方式
 - ☞ 物理与逻辑结构

Oracle的组件

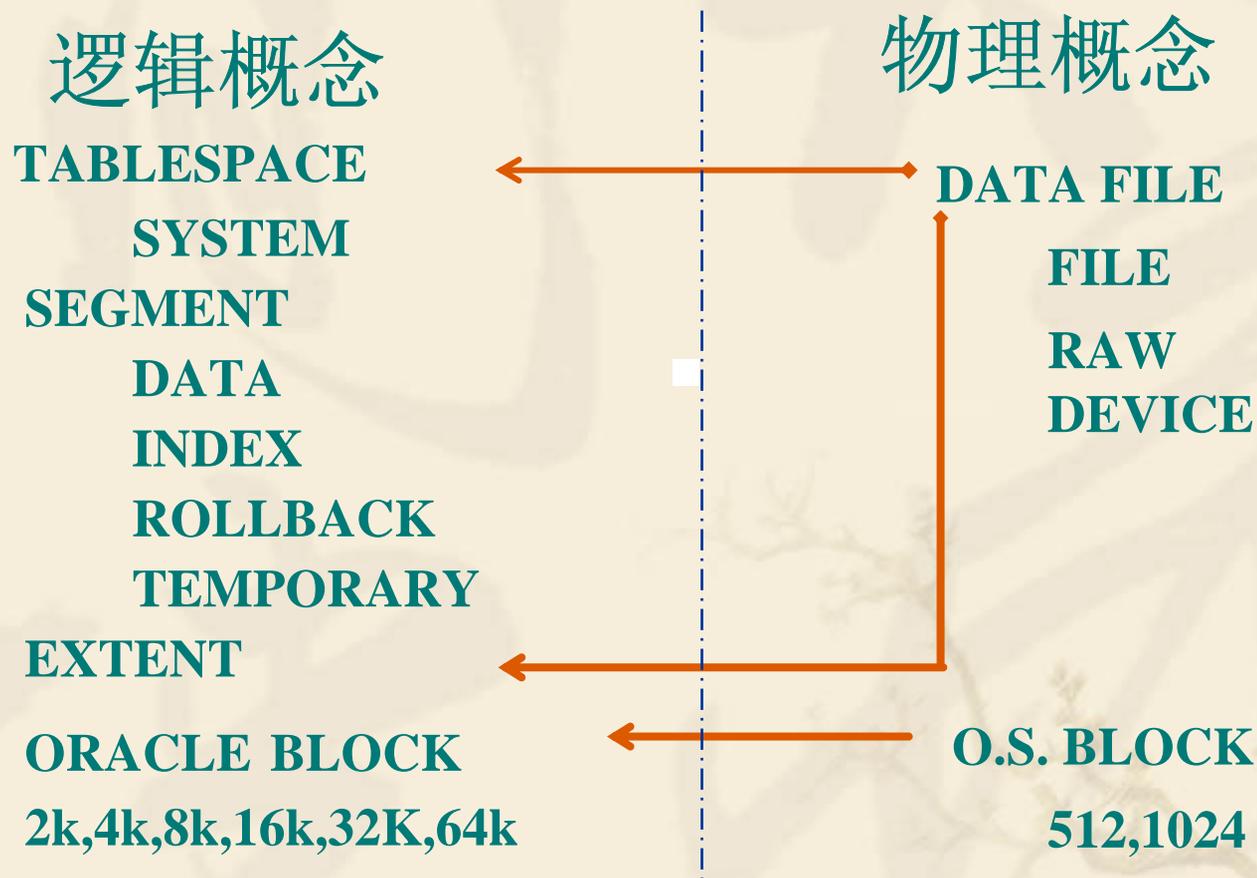
❖ Oracle的组件有：

- ❧ 一组后台进程 组成 一个数据库的Instance
- ❧ 内存结构
- ❧ 一组数据文件与参数文件

Oracle的组件(续1)

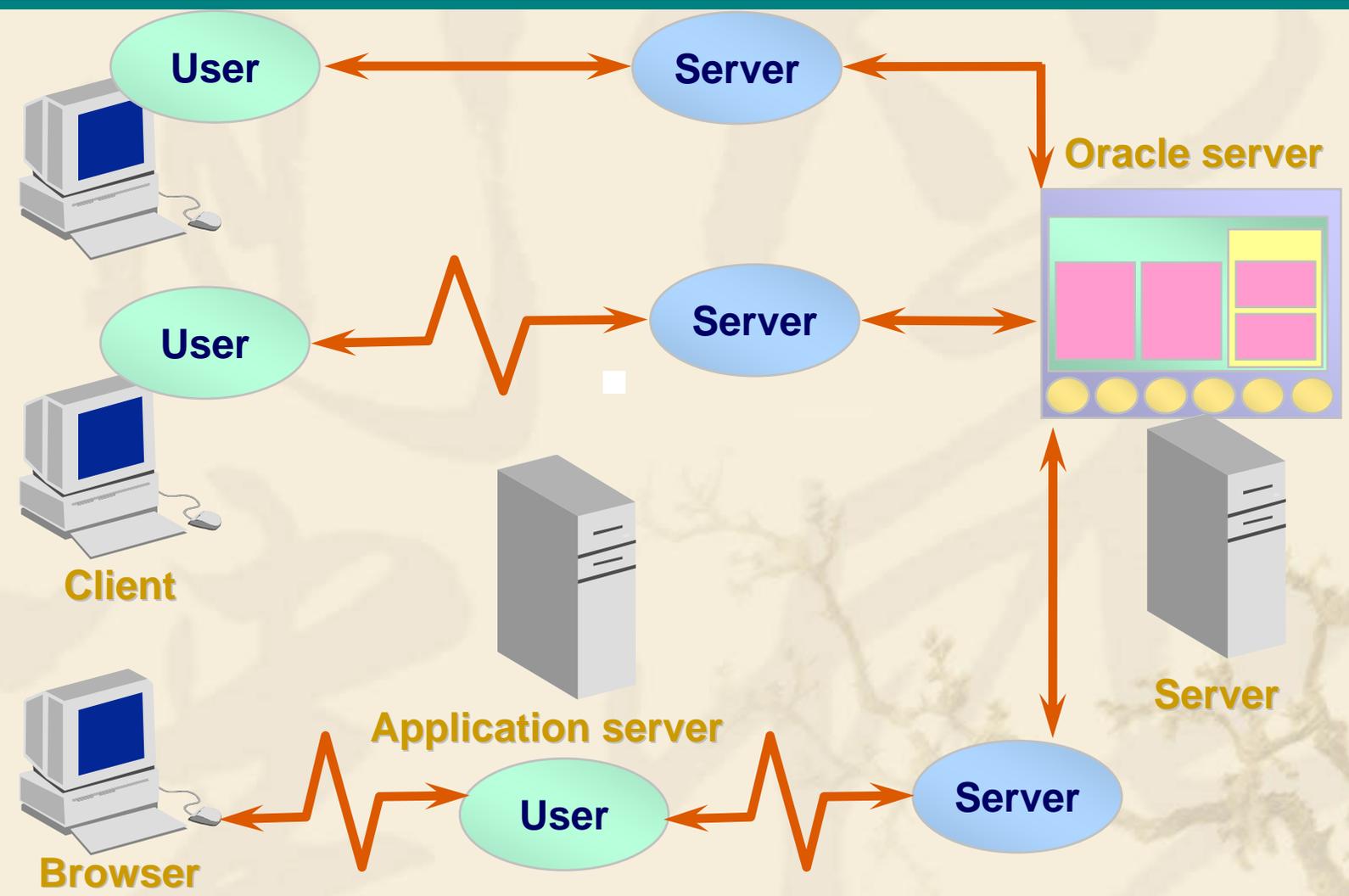


物理与逻辑结构



连接到Oracle Instance的方式

说明:
Oracle
建立一个数据
库的连接是昂
贵的!!!





小结

❖ 本课小结:

☞ Oracle的构架

☞ Instance

☞ SGA

☞ Server Processes

☞ Background Process

☞ 连接到Oracle Instance的方式

☞ 逻辑与物理结构

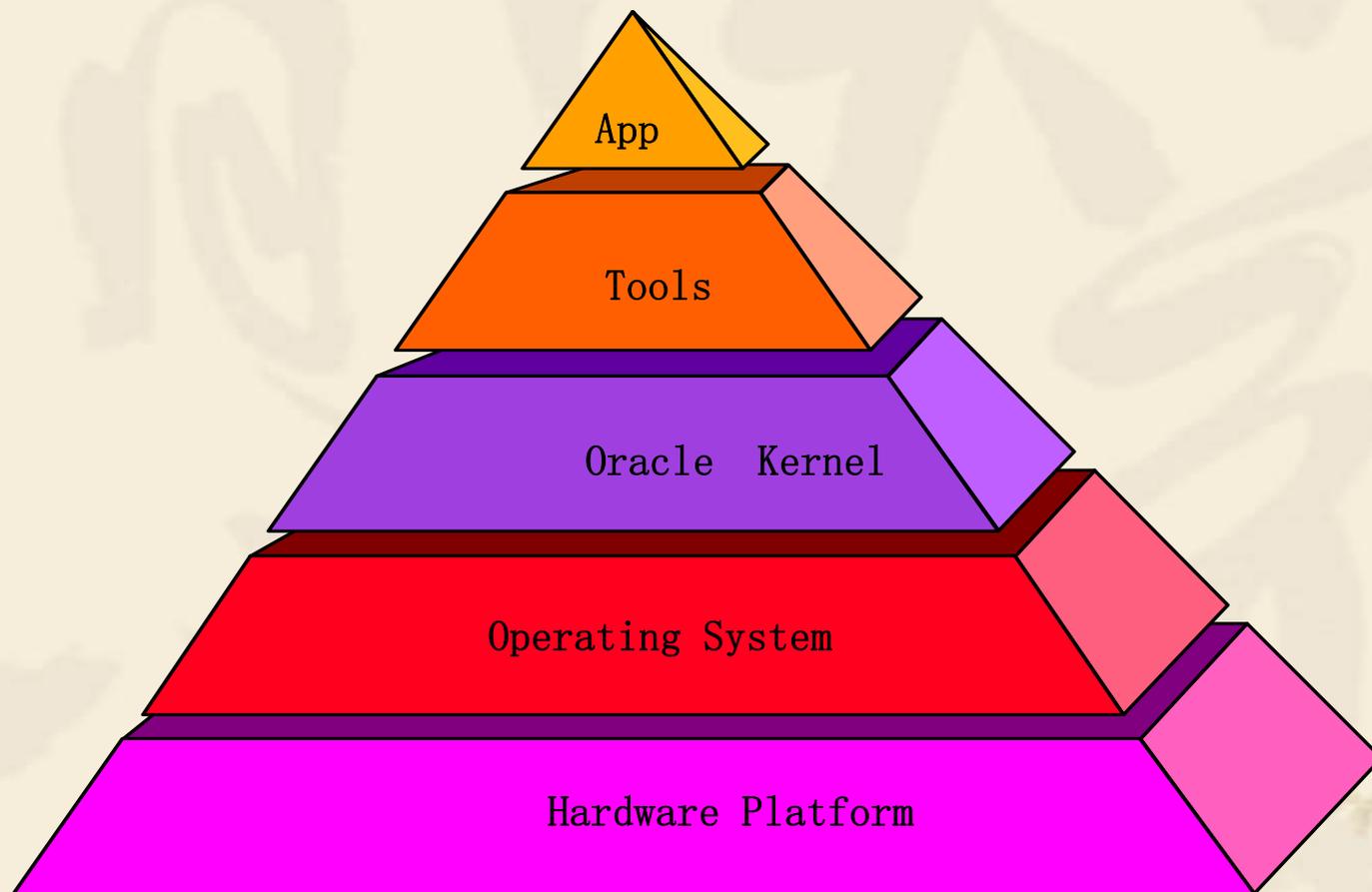


Oracle性能调优概述

❖ 本课将要学习的内容:

- ☞ 应用系统的层次
- ☞ 调优的疑问
- ☞ 调优的目标
- ☞ 调优的方法与瓶颈
- ☞ 调优的步骤

应用系统结构



调优的疑问

❖ 哪些人要做调优工作？

1. 应用设计人员
2. 应用开发人员
3. 数据库管理员
4. 系统管理员

调优的疑问(续1)

❖ 为什么要调优呢?

- ❧ 最好的数据库优化工作是仔细的设计系统与应用
- ❧ 大多数情况下调应用是最主要与有效的优化方法

❖ 对性能影响的几个方面:

- ❧ 硬件系统的设计要适合用户的需求
- ❧ 小心并仔细的设计Oracle的数据库
- ❧ 应用开发人员要写高效的SQL语句

调优的疑问(续2)

❖ 怎么调优?

❧ 首先必须有一个清晰的思想——试着达到

❧ 尽可能的量化你的目标



调优—测量与方法

- ❖ 响应时间
- ❖ 数据库可用性
- ❖ 数据库命中率(百分比)
- ❖ 内存利用率

调优的目标

- ❖ 存取最少的数据块
- ❖ Cache Blocks in memory
- ❖ 共享应用代码
- ❖ 被读写的数据要有尽可能的快
- ❖ 确保用户不等待资源
- ❖ 备份与日常维护对系统最小化

调优的步骤

1. 调设计
2. 调应用
3. 调内存
4. 调IO
5. 调冲突点
6. 调操作系统

小结

❖ 已经学习过的内容:

- ☞ 调优的总体思想
- ☞ 什么要做调化?
- ☞ 什么与性能相关
- ☞ 调优的目标
- ☞ 调优的方法



Oracle表、索引|概念

- ❖ 本课将要学习的内容：
 - ☞ 表空间的概念与性能关系
 - ☞ 表的概念与性能关系
 - ☞ 索引的概念与性能关系



表空间(Tablespace)的概念

- ❖ 是区分数据库存储的逻辑单元
- ❖ 是一组相关的逻辑结构体
- ❖ 表空间由数据文件组成(datafiles)

表空间的分类

❖ SYSTEM表空间:

- ☞ 数据库创建时同时建立
- ☞ 上面存放数据字典
- ☞ 表空间上有系统回滚段(system rollback segment)

❖ Non-SYSTEM表空间:

- ☞ Separate segments
- ☞ 便利的空间管理
- ☞ 控制分配给用户的大量空间



表空间的管理与性能

- ❖ 数据字典管理的表空间:
 - ❧ 这是个默认技术
 - ❧ 表空间中的自由扩展(Free extents)记录在数据字典中
- ❖ 本地管理的表空间:
 - ❧ 表空间中的自由扩展(Free extents) 记录在一个位图中
 - ❧ 每一个位对应一个数据块或一组数据块
 - ❧ 位值的不同表示是空闲块还是使用过的块
- ❖ 本地管理要比数据字典管理的性能要好。



本地管理的表空间特点

- ❖ 减少递归空间(recursive space)管理
- ❖ 减少在数据字典上的资源竞争
- ❖ 不会有回滚段的使用
- ❖ 没有空间接合的需要



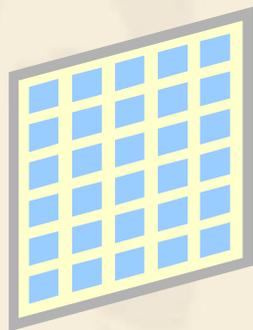
排序与临时表空间

- ❖ 用作排序操作
- ❖ 不能有任何的永久对象中这种表空间上
- ❖ 推荐使用本地管理的表空间
- ❖ $UNIFORM\ SIZE = SORT_AREA_SIZE * n$

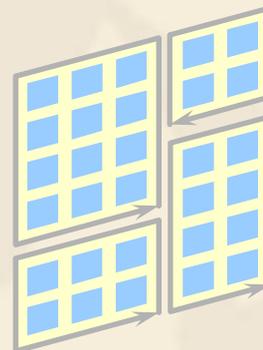
表

- ❖ 表是Oracle中的最基础的数据存储单元
- ❖ 数据被按行(Row)与列(column)的方式存放
- ❖ 表可以做分区来提高性能(800M以上)

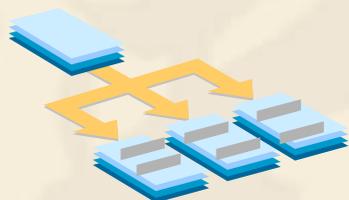
表的种类



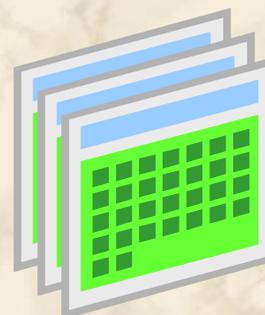
**Regular
table**



**Partitioned
table**

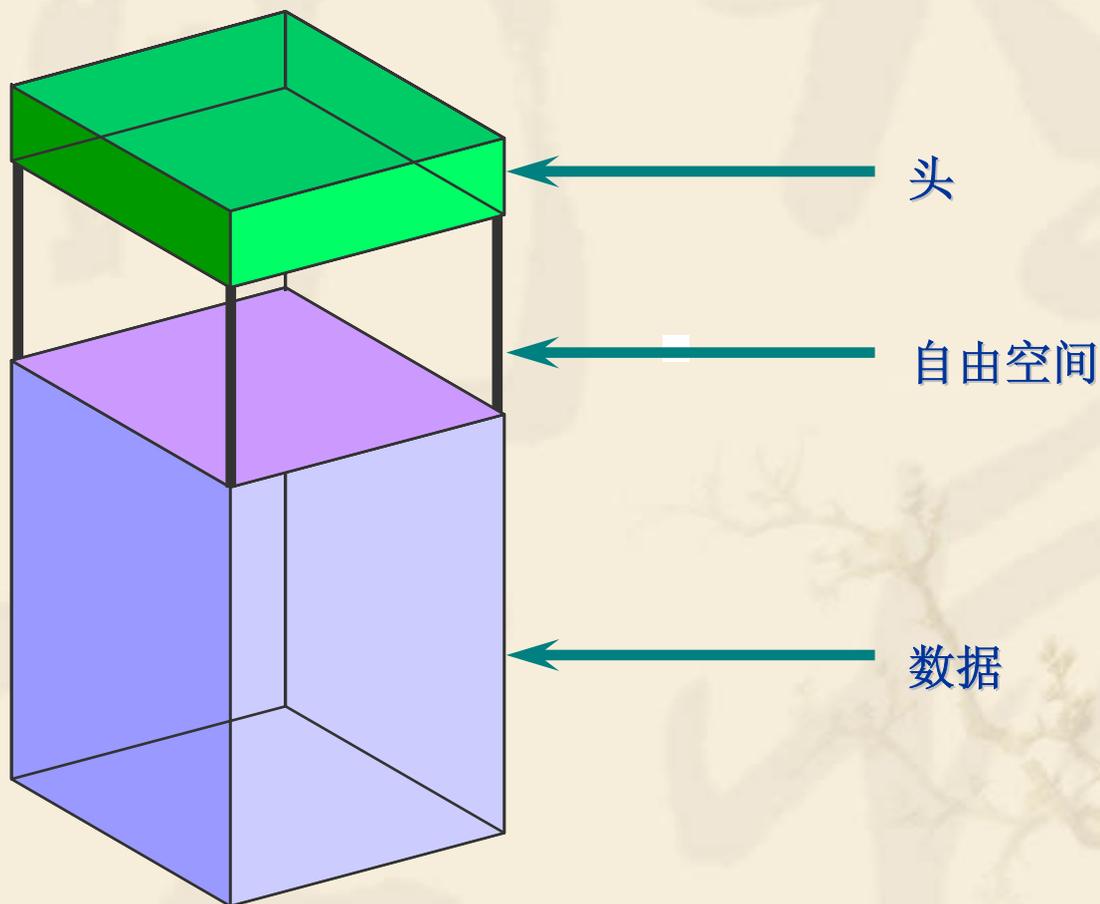


**Index-organized
table**



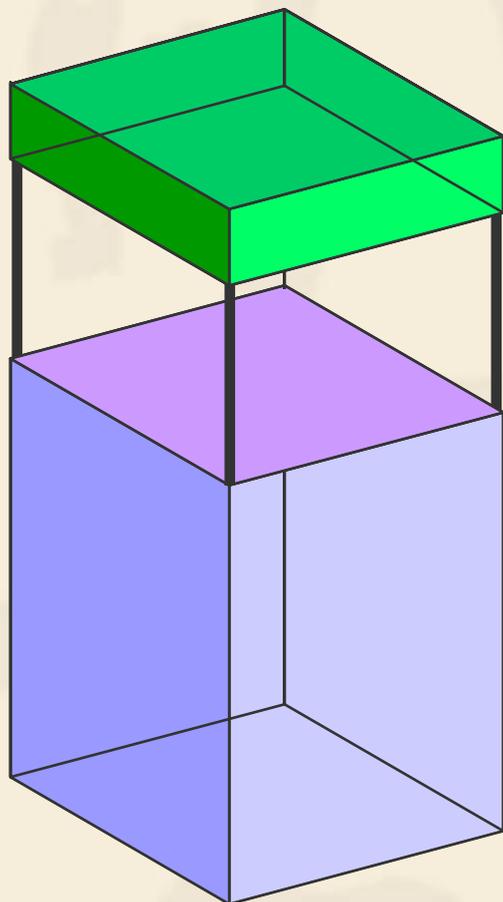
Cluster

表参数说明—数据块结构



❖ 数据块
(block)
的组成

表参数说明—数据块参数



INITRANS

MAXTRANS



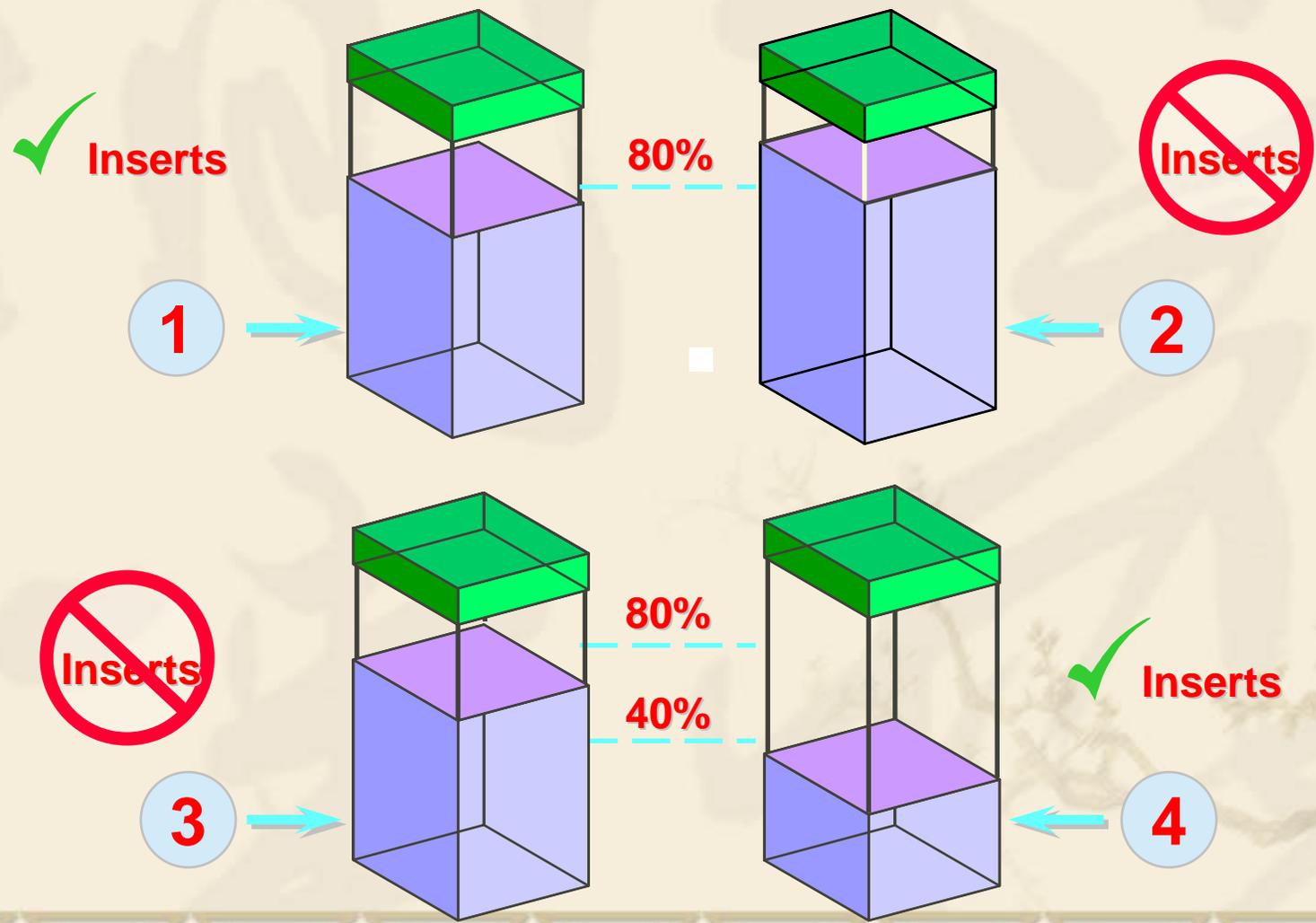
PCTFREE

PCTUSED —Oracle 9i中在使用
Tablespace space
management auto的情况下此参
数无效

数据块空间使用方法

PCTFREE=20

PCTUSED=40





建表指南

- ❖ 使用较少的标准扩展，减少表空间的碎片。
- ❖ 使用本地管理表空间减少碎片。
- ❖ 小表，并且经常使用，在表上使用 CACHE 子句。



表的管理指南

- ❖ 在创建时一定要做设计
 - ☞ 所创建之表是规范化的
 - ☞ 每个列(Column)都有一个正确的数据类型
 - ☞ 列(Column)是否定义为空(NULL), 这样可以节约存储空间
 - ☞ 是否可以使用簇表, 可以节约存储空间并优化SQL执行的性能
- ❖ 指定数据块(data block)空间的使用
 - ☞ 每一张表都要指定PCTFREE、PCTUSED参数



表的管理指南(续1)

❖ 指定每个表的存储位置

- ❧ 在建表时指定Tablespace参数

- ❧ 指定位置可以提高数据库系统性能

- ❧ 可以减少数据库管理时间

❖ 创建表时要考虑并行处理

- ❧ 在使用CREATE TABLE ... AS subquery时使用并行处理可以提高建表的速度



表的管理指南(续2)

- ❖ 创建表时要考虑NOLOGGING参数的使用
 - ☞ 用CREATE TABLE ... AS SELECT 建表时使用NOLOGGING会提高建表时间
 - ☞ 使用NOLOGGING参数是最小化redo信息
 - ☞ SQL*Loader、Direct Load Insert操作不做日志
 - ☞ 接下来的DML语句(UPDATE、DELETE、conventional path insert)不会受该参数的影响，会产生redo log。
 - ☞ 这个参数对大表有益，对小表没有什么明显改善。



表的管理指南(续3)

❖ 估计表的大小并设定存储参数

- ❧ 可以计算出使用的硬件空间，是采购与分配存储的依据
- ❧ 估计大小也是合理设置存储参数的依据，以提高IO性能

❖ 对大表做计划

- ❧ 设定MAXEXTENTS参数
- ❧ 表与索引分离
- ❧ 分配足够的临时段空间



表的管理指南(续4)

❖ 考虑表的限制:

- ❧ 高版本数据库表的新对象类型不能导入到低版本
- ❧ 表大小的限制
- ❧ 表中列(Column)个数的限制

索引概念

- ❖ 索引是表、簇的一个可选的关联结构
- ❖ 一个表可以创建一个或多个列(Column)的索引加速SQL语句的执行
- ❖ 一个表可以有多个索引，一个列也可以建多个索引

索引的类型

1. B-Tree indexes —the default and the most common
2. B-tree cluster indexes—defined specifically for cluster
3. Hash cluster indexes—defined specifically for a hash cluster
4. Reverse key indexes—most useful for Oracle Real Application Cluster applications

索引的类型(续1)

1. Bitmap indexes—compact; work best for columns with a small set of values
2. Bitmap Join indexes
3. Function-based indexes—contain the precomputed value of a function/expression
4. Domain indexes—relate to partitioned tables and indexes
5. Global and local indexes—specific to an application or cartridge



索引使用指南

❖ 创建索引在表的数据导入后

- ❧ 如果一个表经常使用SQL*Loader、Import utility导入数据
- ❧ 表有索引，导入数据时要设置SORT_AREA_SIZE参数准备索引的排序空间

❖ 索引要创建在正确的表与列上

- ❧ 在经常查询的返回结果集小于15%时，针对查询条件的列建索引
- ❧ 在多表做连接(Join)时，使用索引的列做连接可以提高性能
- ❧ 小表不用创建索引



索引使用指南(续1)

❖ 哪些列可以做索引

☞ 具有相关惟一性的列

☞ 值范围很大的列适合做常规索引(regular index)

☞ 值范围很小的列适合做位图索引(Bitmap index)

☞ 列的值大多数的空值，但是经常查询所有的非空值，使用下面的语法：

```
WHERE COL_X > -9.99 * power(10,125)
```

上面的写法优于：

```
WHERE COL_X IS NOT NULL
```

因为第一个会使用到索引(假定COL_X是数字型)



索引使用指南(续2)

- ❖ 哪些列不适合做索引
 - ❧ 列中含有大量的空值，但是又不做非空值的查询
- ❖ LONG 和 LONG RAW 列不能做索引
- ❖ 一个索引的入口 (index entry) 大小不能超过数据块有效空间的一半



索引使用指南(续3)

- ❖ 排索引列的位置顺序可以提高查询的性能
- ❖ 限制在每个表上的索引个数
 - ☞ 表可以有任意多的索引
 - ☞ 过多的索引使用Insert、update、delete操作过慢
- ❖ 删除表上不要的索引
 - ☞ 不能提高查询的速度
 - ☞ 应用不再使用的索引
- ❖ 设定索引的块空间(block space)使用参数
 - ☞ 设定PCTFREE参数, 这个值是为新的数据插入和索引入口(index entry)指定空间



索引使用指南(续4)

- ❖ 评估索引的大小并设定存储参数
- ❖ 指定每个索引的表空间
- ❖ 考虑创建并行索引
- ❖ 创建索引时考虑使用NOLOGGING参数
 - ☞ 减少redo log的开销
 - ☞ 加快索引的创建
 - ☞ 提高并行创建大索引的性能



索引使用指南(续5)

❖ 考虑接合(coalescing)、重建(rebuild)索引的益处与成本

Rebuild Index	Coalesce Index
快速将索引移动到其它表空间	不做表空间的移动
成本高: 要更多的存储空间	成本低: 不要多的存储空间
创建一个新的tree, 如果可能收缩深度	在tree同一个分枝(branch), 合并叶节点(leaf)
在不用删除原索引的情况下快速存储与表空间	快速释放叶节点的空间为以后使用

❖ 考虑删除、禁用(disable)约束(constraint)的成本

本课小结

- ❖ 本课学习到的内容:
 - ☞ 表空间的概念与使用
 - ☞ 表的概念与使用
 - ☞ 索引的概念与使用



Oracle 函数、过程

❖ 本课将要学到内容:

❧ 函数的概念与使用

❧ 过程的概念与使用

函数

- ❖ 函数(function)类似操作符
- ❖ 操作数据项，并且返回一个值
- ❖ 与操作符不同是它有一定格式的输入参数
- ❖ 输入参数的个从0到n个

如: `func(arg1,arg2,.....)`



函数(续1)

❖ 函数分类:

❧ 单行函数(single-row-function)。这类函数可以使用在:

- ❖ Select list
- ❖ WHERE clauses
- ❖ START WITH and CONNECT BY clauses
- ❖ HAVING clauses

❧ 聚合函数(aggregate-function)。可以使用在:

- ❖ Select list
- ❖ ORDER BY clauses
- ❖ GROUP BY clauses

函数(续2)

❖ 函数分类:

- ❧ 分析函数(analytic-function)。用于计算一组行的聚合值，返回多行并于行组对应。主要用于OLAP
- ❧ 对象引用函数(object-reference-function)。用于对象的引用方面的操作。
- ❧ 用户定义函数(user-defined-function)

过程(procedure)

- ❖ 与函数基本一样
- ❖ 有一个名字(在一个schema内要唯一)
- ❖ 可以有参数与返回值
- ❖ 可以被许多人调用(同时或非同时)
- ❖ 存储在数据字典中



PL/SQL的使用说明

- ❖ 在PL/SQL的语句块中可以使用Oracle的内建数据类型
- ❖ 在PL/SQL的语句块(单元)中使用PL/SQL的预定义数据类型可以提高性能, 如: pls_integer的数据类型就要比使用number的整型要快
- ❖ 使用wrap可以将PL/SQL的function、procedure、package的内容隐藏
- ❖ PL/SQL可以编译生成本地代码



PL/SQL编译生成本地代码

- ❖ PL/SQL一般在使用时都是解释(interpretive)方式在执行, 我们可以将它变为本地的C语言来执行。
- ❖ 首先安装Oracle Server计算机上必须有C语言的编译器。
- ❖ 这个过程只是加快PL/SQL的执行, 不能加快SQL的执行。
- ❖ 编译后的PL/SQL生成shared library
- ❖ 这个操作是动态的, 不用重启数据库引擎。



PL/SQL编译生成本地代码 - 配置

- ❖ 在\$ORACLE_HOME/plsql/目录中有个名为spnc_makefile.mk的makefile, 将这个makefile中的“CC”指定到你的C语言编译器。
- ❖ 校验\$ORACLE_HOME/plsql/spnc_makefile.mk文件中的下表中的路径。

PL/SQL编译生成本地代码 - 配置(1)

❖ 在spnc_makefile.mk文件中要做校验的变量

Entry in file	Unix command to verify	Output to Unix command
PLSQLHOME=\$(ORACLE_HOME)/plsql/	ls -d \$ORACLE_HOME/plsql	/oracle/app/oracle/product/9.2.0/plsql
PLSQLINCLUDE=\$(PLSQLHOME)include/	ls -d \$ORACLE_HOME/plsql/include	/oracle/app/oracle/product/9.2.0/plsql/include
PLSQLPUBLIC=\$(PLSQLHOME)public/	ls -d \$ORACLE_HOME/plsql/public	/oracle/app/oracle/product/9.2.0/plsql/public

PL/SQL编译生成本地代码 - 配置(2)

❖ 在spnc_makefile.mk文件中要做校验的变量

Entery in file	Unix command to verify	Output to Unix command
RM=/usr/bin/rm -f	which rm	/usr/bin/rm
CC=/usr/bin/xlc	which xlc	/usr/vac/bin/xlc
LD=/usr/bin/ld	which ld	/usr/bin/ld

PL/SQL编译生成本地代码 - 配置(3)

- ❖ 将 `PLSQL_COMPILER_FLAGS` 参数一定要设为 `NATIVE`。

SQL> ALTER SYSTEM SET plsql_compiler_flags=NATIVE SCOPE=both;

- ❖ 同时还有六个参数要做设置，下面做详细的说明。

PL/SQL编译生成本地代码 - 配置(4)

参数	说明	值
1. PLSQL_NATIVE_LIBRARY_DIR	该参数为目录，存放被编译对象	/oracle/app/oracle/product/9.2.0/plsql/lib
2. PLSQL_NATIVE_C_COMPILER	指定C编译器的全路径。这是一个可选的参数，Oracle可以使用 <code>spnc_makefile.mk</code> 文件中设定的'CC'，如果设置了该参数，Oracle将忽略 <code>spnc_makefile.mk</code> 文件中的设置。	使用默认的 <code>spnc_makefile.mk</code> <code>makefile</code>

PL/SQL编译生成本地代码 - 配置(5)

参数	说明	值
3. PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT	该参数降低在大量的编译对象放在 PLSQL_NATIVE_LIBRARY_DIR 指定的目录中对性能的负面影响。如果有超过10000个编译对象时，最好分开到几个子目录中，这样将存取操作对性能的影响降到最小。 例如： PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT 设为500，这样就有500个子目录，子目录的命名为一定要是‘d’字母开头，d0到d499	当值设为500时： /oracle/app/oracle/product/9.2/plsql/lib/d0 到 /oracle/app/oracle/product/9.2/plsql/lib/d499

PL/SQL编译生成本地代码 - 配置(6)

参数	说明	值
4. PLSQL_NATIVE_LINKER	指定链接器的全路径，这个参数也是可选的，如果设置将覆盖 spnc_makefile.mk 文件中的设置	Use default in spnc_makefile.mk makefile
5. PLSQL_NATIVE_MAKE_FILE_NAME	设定makefile的带文件名的全路径，默认值是 /oracle/app/oracle/product/9.2/plsql/spnc_makefile.mk	/oracle/app/oracle/product/9.2/plsql/spnc_makefile.mk
6. PLSQL_NATIVE_MAKE_UTILITY	本地的make应用带文件名的全路径。	/usr/bin/gmake

PL/SQL编译生成本地代码 - 编译

- ❖ 以上的配置好后就可以做编译操作：
 - ☞ 通过CREATE一个procedure、package就可以做到
 - ☞ 如果是已经存在的procedure、package就可以使用ALTER [PROCEDURE | PACKAGE] {procedure_name|package_name} COMPILE 命令来实现编译生成本地化代码
- ❖ 编译成本地代码后将提高程序的跳转、函数调用的速度，优化内存的使用。

本课小结

- ❖ 函数知识
- ❖ 过程知识
- ❖ PL/SQL的使用与PL/SQL的本地编译方法

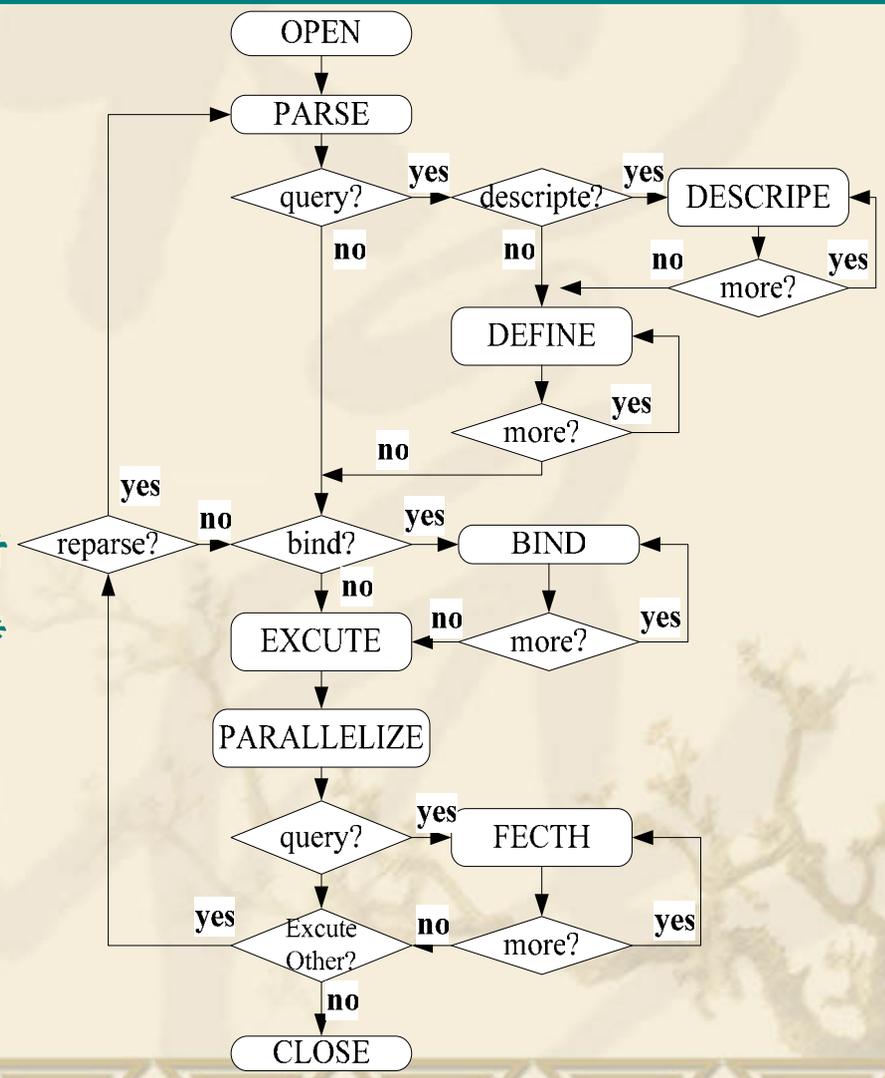
Oracle的SQL语句的处理过程与事务

❖ 本课学习内容:

- ❧ Oracle是如何处理SQL语句
- ❧ Oracle8i后的新事务控制

Oracle的SQL处理过程

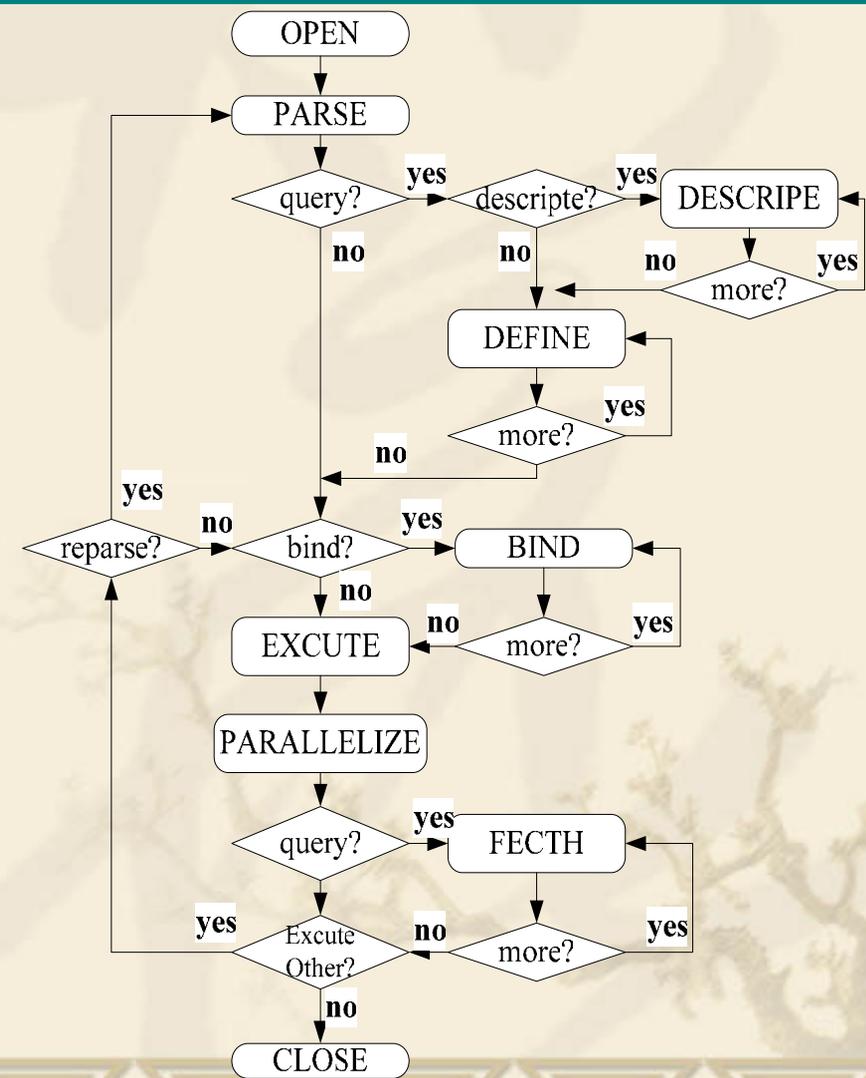
1. Oracle Server接收到SQL语句后
第一步是做分析(语法、对象依赖关系的检查)
2. 判断是否为查询语句?
3. 是否有变量的绑定?
4. 对语句做执行
5. 如果有并行参数的设置做并行执行
6. 执行出结果后再判断是否为查询语句?



Oracle的SQL处理过程(续1)

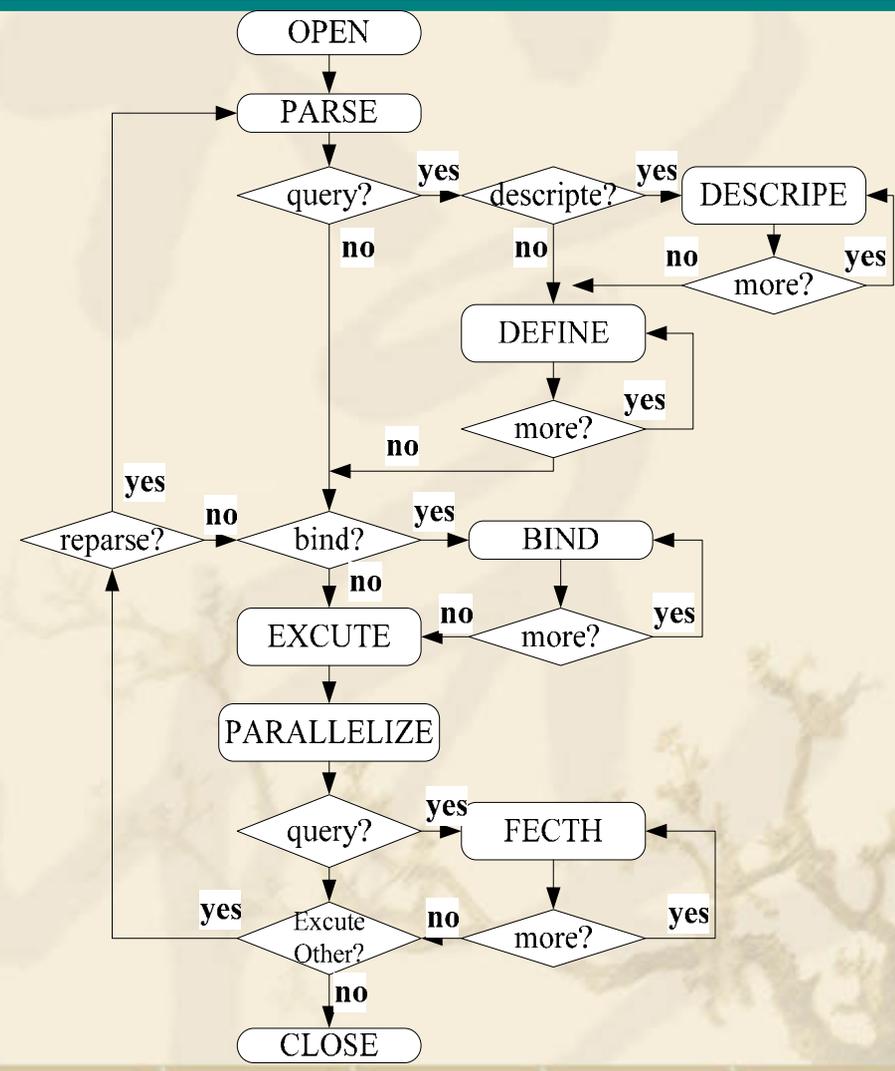
❖ Parsing, 是Oracle接收到应用的SQL要做的第一件事。Parsing做如下的内容:

- ❧ 检查语法与语义的有效性
- ❧ 查看这个发出的SQL是否有权限执行
- ❧ 为语句分配私有的SQL区域 (private SQL area)
- ❧ 判断是否已经有该语句存在 Shared SQL area



Oracle的SQL处理过程(续2)

- ❖ Stage 1: Create a Cursor
- ❖ Stage 2: Parse the Statement
- ❖ Stage 5: Bind Any Variables
- ❖ Stage 7: Run the Statement
- ❖ Stage 9: Close the Cursor





Oracle事务处理

- ❖ 事务是一个逻辑工作单元所需要的所有内容的组合。
- ❖ 将一组相关的SQL操作放到一个事务中。
- ❖ Oracle的事务可以使用commit做提交，使用rollback做回滚，使用savepoint做事务保存。

ANSI 隔离级别(isolation levels)

Isolation levels	Dirty Read(1)	Non-Repeatable Read(2)	Phantom Read(3)
Read Uncommitted	Possible	Possible	Possible
Read committed	Not Possible	Possible	Possible
Repeatable Read	Not Possible	Not Possible	Possible
Serializable	Not Possible	Not Possible	Not Possible
Notes:	<ol style="list-style-type: none">1. A transaction can read uncommitted data changed by other transaction2. A transaction rereads data committed by other transaction and sees the new data.3. A transaction can execute a query again, and discover new rows inserted by other committed transaction		

Oracle在ANSI 隔离级别的说明

Isolation levels	Description
Read Uncommitted	Oracle never permits "dirty reads." Although some other database products use this undesirable technique to improve throughput, it is not required for high throughput with Oracle.
Read Committed	Oracle meets the READ COMMITTED isolation standard. This is the default mode for all Oracle applications. Because an Oracle query only sees data that was committed at the beginning of the query (the snapshot time), Oracle actually offers more consistency than is required by the ANSI/ISO SQL92 standards for READ COMMITTED isolation.
Repeatable Read	Oracle does not normally support this isolation level, except as provided by SERIALIZABLE.
Serializable	You can set this isolation level using the SET TRANSACTION command or the ALTER SESSION command.



Oracle 8i后的新事务控制

- ❖ 在Oracle8i以前的数据库中，没有办法将一些跟踪或一定要记录的信息直接写到表中保存下来，必须先写到数组这样的变量中，在主事务commit或rollback后，再将程序中变量保存的数据写到表中。
- ❖ 在Oracle8i后，提供了AUTONOMOUS TRANSACTION(自治事务，以下称AT)



什么是AUTONOMOUS TRANSACTION

- ❖ Autonomous transaction(AT)是一个独立的事务，并且它一定是被主事务（main transaction - MT）来启动
- ❖ 当AT启动时，将会使用MT挂起，即暂停运行，AT这时可以执行自己的SQL操作，可以将这些SQL操作commit,rollback，AT运行完成后继续MT的运行。
- ❖ AT使用简单，在程序使用“pragma AUTONOMOUS_TRANSACTION”做AT事务的开始，使用commit或rollback结束AT事务。

AT的使用方法

- ❖ AT可以在下面的程序中运用
 - ❧ 顶级(Top-level)的PL/SQL块中
 - ❧ 本地的、单独的函数(function)或过程(procedure)
 - ❧ 包(package)中的函数或过程
 - ❧ SQL的对象类型 (Object types) 的方法中
 - ❧ PL/SQL中的触发器 (trigger)

AT的调用示例

Main Routine

```

PROCEDURE procl IS
  emp_id NUMBER;
BEGIN
  emp_id := 7788;
  INSERT ...
  SELECT ...
  proc2;
  DELETE ...
  COMMIT;
END;

```

Autonomous Routine

```

PROCEDURE proc2 IS
  PRAGMA AUTON...
  dept_id NUMBER;
BEGIN
  dept_id := 20;
  UPDATE ...
  INSERT ...
  UPDATE ...
  COMMIT;
  INSERT ...
  INSERT ...
  COMMIT;
END;

```

MT begins

MT ends

MT suspends

AT1 begins

AT1 ends

AT2 begins

AT2 ends

MT resumes

AT的一些特点

- ❖ 改变AT的结果不影响MT最终的部署与状态，如：
 - ⌘ AT看不到任何被MT修改的数据
 - ⌘ 当AT中做事务的提交与回滚，不会影响外层MT的事务控制
- ❖ 当AT中做了事务的提交，其它的事务就能看到数据的变更
- ❖ 一个AT可以启动另一个AT

本课小结

- ❖ 讲述了Oracle的SQL处理过程
- ❖ 对Oracle8i以后的AT是什么，如何工作的做了介绍



Oracle的Optimizer学习

- ❖ 本课学习的内容:
 - ☞ SQL 处理的构架
 - ☞ Optimizer的介绍
 - ☞ 学习执行计划(execution plan)
 - ☞ 如何选择一个Optimizer方法(Approach)与目标(Goal)
 - ☞ 可以扩展的优化器(Extensible Optimizer)介绍
- ❖ 说明: RBO是以Oracle8i版本为基础, CBO是以Oracle9i为基础

SQL 处理的构架 —— 组件介绍(一)

- ❖ SQL处理构架包括的主要组件有：
 - ☞ 分析器 (Parser)
 - ☞ 优化器 (Optimizer)
 - ☞ 行源生成器 (Row source Generator)
 - ☞ SQL执行的引擎 (SQL execution Engine)

SQL 处理的构架 —— 组件介绍(二)

❖ 分析器(Parser)

☞ 分析器主要有两个功能:

- ❖ 语法分析

- ❖ 语义分析, 例如查检被使用到的数据库中的对象和对象属性是不是正确

SQL 处理的构架 —— 组件介绍(三)

❖ 优化器(Optimizer)

- ❧ 优化器是使用数据库内部的规则或成本方法，确定一个最有效的将查询结果返回的路径。
- ❧ 优化器输出一个描述最优化的执行方法的计划(plan)，
- ❧ Oracle提供两种优化器：RBO – 基于规则，CBO – 基于成本，在Oracle9i以后推荐使用CBO，在Oracle10g中不再支持RBO。

SQL 处理的构架 – 组件介绍(四)

❖ Row source generator – 行源生成器

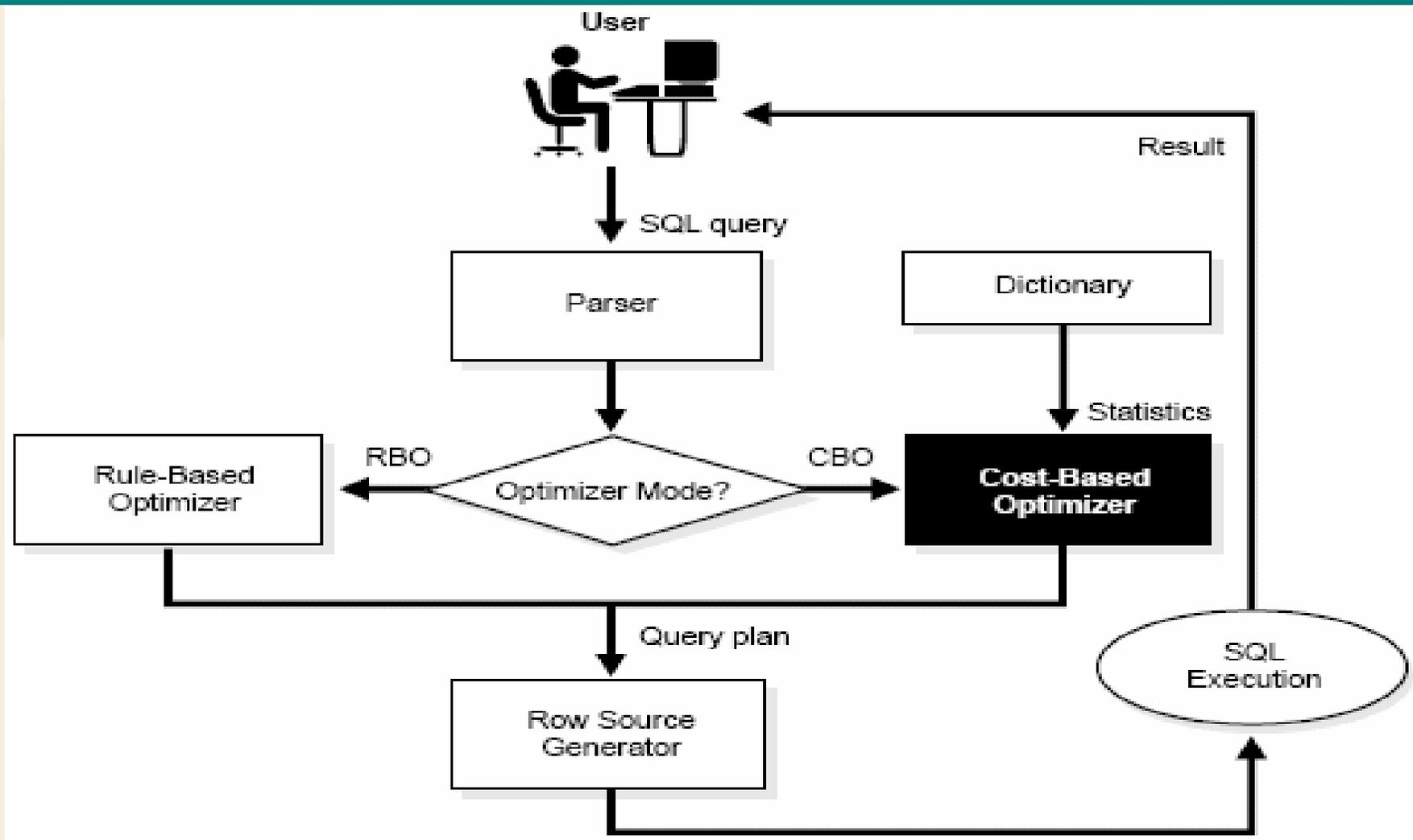
- ❧ 它接收从优化器来的最优计划,
- ❧ 输出SQL的最优执行计划(execution plan),
- ❧ 执行计划以树形结构表示的一组结构化的行源, 每个行源将返回在那一执行阶段(step)的数据行结果集

SQL 处理的构架 —— 组件介绍(五)

❖ SQL Execution Engine

- ❧ SQL 的执行引擎是将执行计划与SQL语句关联起来的操作组件，它将生成查询的结果，
- ❧ 行源生成器生成的每个行源在SQL的执行引擎中被逐一执行。

SQL 处理的构架



Optimizer介绍

- ❖ 优化器是使用数据库内部的规则或成本方法，确定一个最有效的将查询结果返回的路径。
- ❖ 通常一个SQL可能会有多种不同的执行路径，如表和索引不同的存取顺序，这个对Oracle的一个语句执行时间有很大的影响。
- ❖ 在一个查询语句中，优化器会考虑多种被引用的对象与特定的一些条件，优化器会选择基于规则或成本的路径
- ❖ 改变执行计划可以通过设置优化器的方法(approche)或目标(goal)，在CBO中，重新收集代表性的统计数据，开发人员使用优化器的暗示(hints)
- ❖ **注意：Oracle从一个版本升级到另一个版本，可能不会使用同样的执行计划!!!**



Optimizer操作步骤

操作	说明
1、计算表达式与条件	优化器第一步就是尽可能全的计算表达式求值，并判断条件
2、语句变换	对于复杂的语句，如关联子查询或视图，优化器将原始的语句转换为连接(Join)语句。
3、选择优化器的路径	选是基于规则的还是成本的，并确定优化目标(goal)
4、选择存取路径	语句中每个存取的表，优化器选择一个或多个有效的存取表数据的路径
5、选择连接的顺序(Join orders)	超过两个以上的表做连接时，优化器要判断哪两个表先做连接，然后再使用结果集连接下一个表
6、选择连接方法	对于任意的连接语句，优化器要选择一个连接的操作去做连接



执行计划(execution plans)

- ❖ 执行一个DML语句Oracle可能要做很多步，每一步都要做从数据库的物理数据中提取行或者使用某种方法来准备。
- ❖ 这些步骤的组合Oracle使用它去执行一个语句，这就是执行计划(execution plans)。
- ❖ 一个执行计划包括对语句中的每个表的存取路径(access path)和表的读取顺序(Join order)与适当的联合方法(join method)。



执行计划 – 如何解释计划

- ❖ Oracle提供了分析每个DML的语句是有怎么样的执行计划的方法：解释执行计划(Explain plan),语句如下：
 - ❧ Explain plan for <DML>
- ❖ 在使用Explain plan前必须要先建立表PLAN_TABLE。



解释计划示例

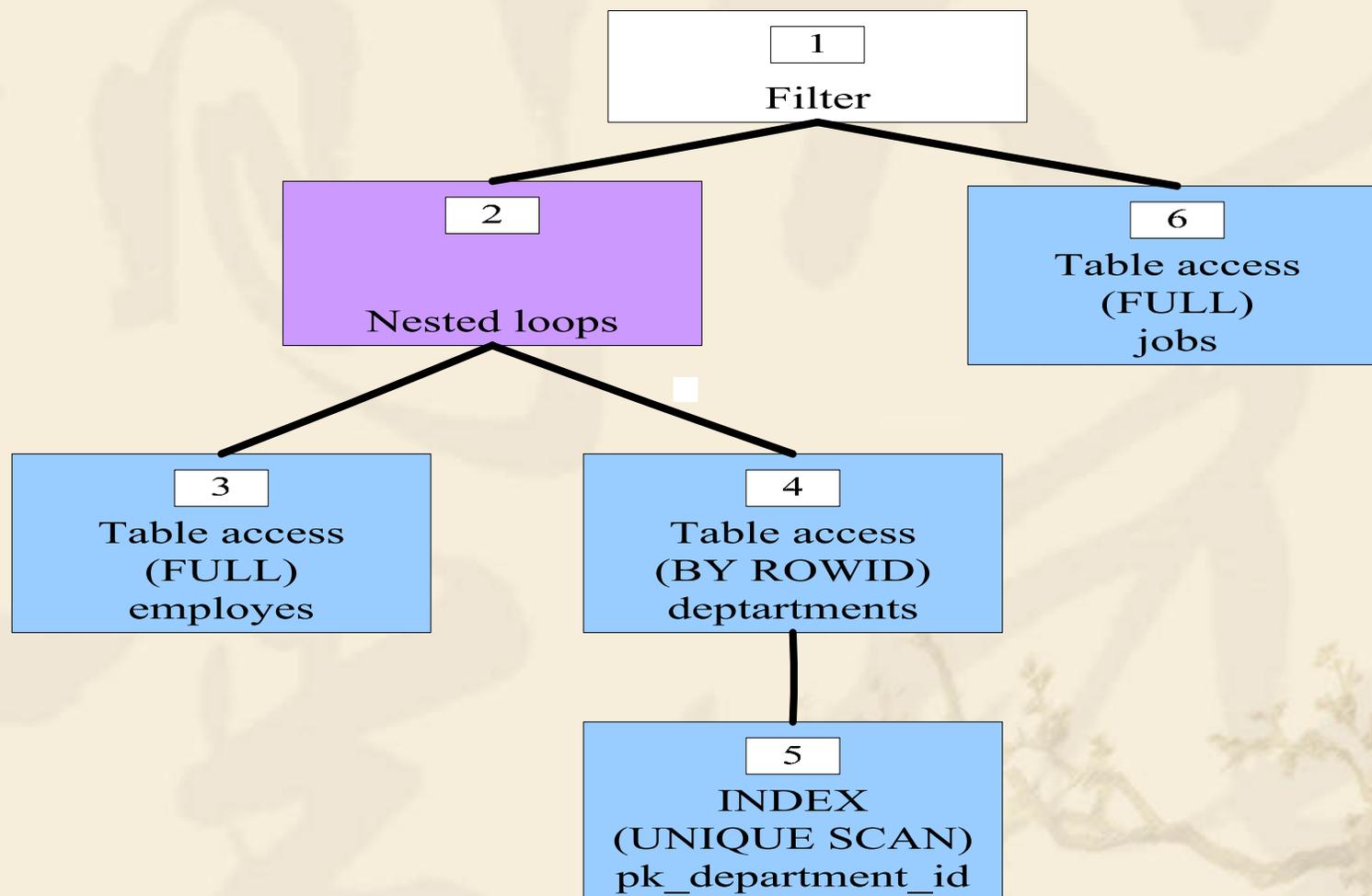
在Sqlplus中运行:

```
EXPLAIN PLAN FOR
SELECT employee_id, job_id, salary, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND NOT EXISTS
  (SELECT *
   FROM jobs
   WHERE employees.salary BETWEEN min_salary AND max_salary);
```

结果:

ID	OPERATION	OPTIONS	OBJECT_NAME
0	SELECT STATEMENT		
1	FILTER		
2	NESTED LOOPS		
3	TABLE ACCESS	FULL	EMPLOYEES
4	TABLE ACCESS	BY ROWID	DEPARTMENTS
5	INDEX	UNIQUE SCAN	PK_DEPARTMENT_ID
6	TABLE ACCESS	FULL	JOBS

解释计划示例说明





选择一个优化器的方法与目标

- ❖ CBO的默认的情况下是获得最大的吞吐量(throughout)
- ❖ Oracle可以优化一个语句得到最佳的响应时间(Best response time)
- ❖ 优化器的目标可以改变一个语句的执行计划，如使用提高吞吐量目标，优化器使用sort_merge操作，如使用最佳的响应时间目标，优化器使用nested_loop操作。



影响优化器工作的因素

- ❖ OPTIMIZER_MODE 初始化参数
- ❖ OPTIMIZER_GOAL 参数，该参数可以使用 alter session 命令来改变
- ❖ 在数据字典中的CBO的统计值
- ❖ 使用Hints

OPTIMIZER_MODE参数设置

值	说明
Choose	<p>1、SQL语句中的使用的至少一个表有统计值，使用最大吞吐量目标的基于成的方法，如果被存取到的对象统计值不完，出来的执行计划是未优化到位的。</p> <p>2、SQL语句中被存取到的任何对象都没有统计值，使用基于规则的优化方法</p>
All_rows	对所有的SQL语句做基于成本的方法做分析与执行，不论使用到的对象是否有统计值，都按最大的吞吐量的方式执行。
First_rows_n	同样使用基于成本的方法来做执行计划，不论被引用到的对象是否有统计值，n的值可以是：1,10,100,1000
First_rows	优化器使用基于加启发式的(heuristic)方式生成最佳的返回头几行结果的执行计划，
Rule	不考虑SQL语句引用对象是否有统计值，对所有的SQL做基于规则的分析



OPTIMIZER_GOAL参数说明

- ❖ 使用ALTER SESSION命令做修改
- ❖ 这个参数只对使用ALTER SESSION命令后的session起作用
- ❖ Optimizer_goal参数的值与optimizer_mode一样
- ❖ 设置optimizer_goal后optimizer_mode参数不再起作用

Oracle的hints

- ❖ 任何hints优先级都高于使用 optimizer_mode、optimizer_goal参数的设置
- ❖ Hints可以使用到的值：
 - ❧ First_rows(*n*)
 - ❧ First_rows
 - ❧ All_rows
 - ❧ Choose
 - ❧ Rule



统计值的收集

- ❖ Oracle 的统计信息是放在数据字典中，使用下面两种方式之一来收集：
 - ⌘ 使用ANALYZE语句，在Oracle 8i以前只能使用它
 - ⌘ DBMS_STAT包，在Oracle 8i以后可以使用它
- ❖ 使用DBMS_STAT可以有更多的灵活性：
 - ⌘ 并行收集
 - ⌘ 收集全局、部分对象统计信息

本课小结

- ❖ 学习了Oracle是如何处理SQL
- ❖ 什么是Oracle的Optimizer
- ❖ 什么是执行计划(Execution plan)
- ❖ 怎么选择优化器的方法(Approach)与目标(goal)

基础篇完

完
高级篇待续