[作者：于海洋]

[循序渐进玩转Oracle数据库，用实例一点一点的渗透知识点。可以满足开发人员的使用]

[Oracle数据库]

个人知识点总结

**Oracle**

***第一部分 数据库常用命令及操作基础***

**如何学习Oracle**

Oracle是目前最流行的数据库之一.功能强大,性能卓越.所以学习比较困难,我个人认为:Oracle完全代表数据库技术的最高水平,因此学习Oracle需要你有一定基础:

1. 学习过一门编程语言(Java,C,C++均可)。
2. 最好学习过一门别的数据库(sql server,MySQL,access)。

严禁没有任何基础的同志学习Oracle,那样你将会浪费时间,同时还打击学习Oracle的信心.我个人认为学习流程为:

1. 计算机基础(了解它的基本原理)
2. 学习一门编程语言(Java,C,C++).
3. 学习Oracle数据库或是别的数据库.

**[注释]-----**如果只会Oracle数据库是不能找到一份工作的.

虽然目前有软件公司招聘dba,但实际上一个只会Oracle,而没有任何项目经验的dba是不可能成为一个合格的dba的,不能根据实际情况对数据库作出相应的处理.

数据库要在项目中才具有价值,不论是数据库的安全性处理,数据库的备份方案的制定,用户的管理等,都是如此.

因此,真正学好Oracle数据库,至少掌握一门语言(Java,C,C++),并做几个项目才能深刻理解Oracle的精妙之处.

**[学习忠告]**------欲速则不达,做任何事都要遵循它的规律,循序渐进.

**【如何学习该课程】**

该课程是基于案例教学的,驱动项目的方式来教授的,因此,是可以比较轻松的掌握的,每一部分我都会带大家完成一个综合案例.

另外,听该课程是可以完全抛开书籍的,只要坚持听下去,完全可以掌握Oracle技术.。

先讲基础部分，把各个知识点搞清楚，在讲解高级部分，综合运用各个知识点，达到融会贯通。整个过程是循序渐进的。【轻松学习，高效掌握Oracle技术】

成为一个Oracle高手

**【学习忠告】**

手的秘诀就是：理解小的知识点→做小练习→把小的知识点连成线（知识线）→做Oracle项目→把知识线形成知识面→深刻理解

***信心很重要。***

**【内容介绍】**

* 基础部分：Oracle基本使用，Oracle用户管理，Oracle表管理
* 高级部分：表的查询，Oracle的权限、角色，Oracle的函数，pl/sql编程，Oracle数据库管理，索引、约束和事务。
* 推荐书籍：Oracle实战教程 从入门到精通；深入浅出Oracle dba入门进阶与诊断案例。

**【内容介绍】**

1. 为什么要学习Oracle
2. 介绍Oracle及Oracle公司的背景
3. Oracle的安装，启动及卸载
4. Oracle开发工具介绍
5. sql\*plus的常用命令
6. Oracle用户的管理

**【期望目标】**

1. 学会安装/启动/卸载Oracle
2. 使用sql\*plus工具
3. 掌握Oracle用户管理
4. 学会Oracle中编写简单的select语句

**【为什么要学习Oracle】----性能优越----薪水高----需求量大**

目前主流数据库包括

* \*微软 sql server和access
* \*瑞典Mysql AB公司mysql
* \*ibm公司 db2
* \*美国Sybase Sybase
* \*ibm公司 informix
* \*美国oracle公司 oracle

数据库分类

* 小型数据库：access，foxbase
* 中型数据库：mysql，sql server，informix
* 大型数据库：sybase，Oracle，db2

**【如何选用数据库】**

1. 项目的规模

|  |  |  |  |
| --- | --- | --- | --- |
| 项目规格 | 小型数据库 | 中型数据库 | 大型数据库 |
| 负载量多大，用户多少 | 小于100人 | 日访问量5000~15000人 | 可以处理海量负载 |
| 成本 | 千元内 | 万元内 | 成本在万元以上 |
| 安全性 | 安全性要求不高 | 安全性一般 | 安全性很高，相对比较贵 |

1. 功能
2. 并发性要求【最终用户】

【dba认证】

Oracle的认证有：dba认证、Java开发认证、Oracle网络开发认证。【但是只有第一个有价值含量】

dba认证：oca[Oracle初级认证]、ocp[Oracle专家级认证]、ocm[Oracle大师级认证]

【Oracle公司介绍】

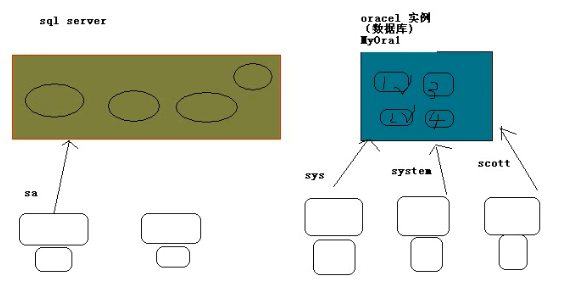
|  |  |
| --- | --- |
| Oracle含义  神喻 代神说话的人 甲骨文。  暗示 公司欲做霸主的决心。  地位：  全球第一大数据库厂商。  全球第二大独立软件供应商。  版本信息   * 1970 数据库起家 * 1980 ORACLE6版本 * 1990 ORACLE7版本，向多元化产品进军，以数据库为优先发展方向。 * 1995 ORACLE8版本 * 1999 ORACLE8i版本 * 2002 ORACLE9i版本 * 2003 ORACLE10g版本 * ...... | Larry Ellisin   * 财富榜前30名之内 * 与比尔盖茨可以相比 * ORACLE的建立者和发展者 * IT风云人物   产品线   * 工具   + Jdeveloper --------JBuilder   + Developer2000 --VB   + Designer2000 ---数据库建模 * 数据库   + ORACLE10g 数据仓库（data warehouse） * 应用软件   + ERP(企业资源计划)   + ORACLE APPLICATION 11i |

【Oracle的安装、启动及卸载】

。。。。。。

【启动】

启动Service和lisener服务



【Oracle的安装、启动及卸载】----删除

运行Oracle的universal installer 。。。。。

【Oracle管理工具的介绍】

【概述】是Oracle自带的工具软件，主要用于执行sql语句，pl\sql块。

如何使用：基于win7的开始目录

1. 在开始——>程序——>Oracle 目录——>application development——>sql\*plus
2. 在运行栏中输入：sqlplusw即可
3. 在开始——>程序——>Oracle目录——>application development——>sql\*plus worksheet

【概述】sqlplus dos 下操作Oracle的工具，其功能和sql\*plus相似。

1. 在运行栏中输入sqlplus。
2. 找到该可执行文件sqlplus.ext，在Oracle主目录\ora90\bin\plussqlplus.exe,鼠标双击就可以

【概述】Oracle的企业管理器（oem oracle enterprise manger）

位置在 开始——>程序——>oracle oraclehome90——>enterprise manager console 即可启动Oracle的企业管理器，是一个图形界面环境。

【概述】pl/sql developer 属于第三方软件。

主要用于开发，测试，优化 Oracle pl/sql的存储过程。比如：触发器，此软件Oracle不带，需要独立安装。

**【sql\*plus常用命令】**

* 连接命令

1. conn[ect]

用法：conn用户名/密码@网络服务名[暗示sysdba/sysoper]，当用特权用户连接时，必须带上 as sysdba 或是 as sysoper

1. disc[onnect]

说明：该命令用来断开与当前数据库的连接

1. passw[ord]

说明：该命令用于修改用户的密码。如果要修改其他用户的密码，需要用sys/system登录。

1. show user

说明：显示当前用户名。

1. exit

说明：该命令会断开与数据库的连接，同时会退出sql\*plus。

* 文件操作命令

1. start和@

说明：运行sql脚本

【案例】sql>@d:\a.sql或者sql>START d:\a.sql

1. edit

说明：该命令可以编辑指定的sql脚本

【案例】sql>edit d:\a.sql

1. spool

说明：该命令可以编辑指定的sql\*plus屏幕上的内容输出到指定的文件中去。

【案例】sql>spool d:\b.sql 并输入sql>spool off

* 交互式命令

1. &

说明：可以替代变量，而该变量在执行时，需要用户输入。

【案例】sql>select \* from emp where job=’&job’

1. edit
2. spool

* 显示和设置环境变量

【概述】可以用来控制输出的各种格式，set show 如果希望永久的保存行管的设置，可以去修改 glogin.sql脚本。

1. linesize

说明：设置显示的行的宽度，默认是80个字符

sql>show linsize

sql>set linesize 90

1. pagesize

说明：设置每页显示的行数，默认是14

用法和linesize一样

***【至于其他环境参数的使用也是大同小异】***

**【Oracle用户的管理】**

1. 创建用户

【概述】在Oracle中要创建一个新的用户使用create user 语句，一般是具有dba（数据管理员）的权限才能使用。

【案例】create user username identified by userpassword;

？创建用户并指定密码

**SQL> create user yuhaiyang identifiedby bruceyu;**

*--创建一个用户 名为：2016 密码：2016*

*--在SQL语言中，""表示的是强调*

**SQL> create user"2016"identifiedby"2016";**

**SQL> grant dba to"2016"**

1. 给用户修改密码

【概述】如果给自己修改密码可以直接使用sql>password用户名；如果给别人修改密码则需要具有dba的权限，或是拥有alter user的系统权限。

【案例】sql>alter user 用户名 identified by 新密码。

1. 删除用户

【概述】一般以dba的身份去删除某个用户，如果用其他用户去删除用户具有 drop user的权限。

【案例】drop user 用户名 【cascade】

【特别说明】在删除用户时，如果要删除的用户，已经创建了表，那么就需要再删除时，带一个参数 cascade ，把属于该用户的所有东西全部删除。

1. 用户管理的综合案例

【概述】创建的新用户是没有任何权限的，甚至连登录到数据库的权限都没有，需要为其指定相应的权限。给一个用户赋权限使用命令grant，收回权限使用命令 revoke。

【案例】grant connection to 用户名

1. 初识权限：权限分为系统权限和对象权限

【系统权限】用户对数据库的相关权限。

【对象权限】用户对其他用户的数据对象（指用户创建的表、视图、存储过程、触发器等一系列的数据库的对象）操作的权限。

对象权限：包括select，insert，update，delete，all，create index......

1. 角色：指Oracle为了赋权方便而设定的将部分权限打包，进行批赋权操作。

角色分类：分为预定义角色和自定义角色。

【案例】connect：就是一个角色，它具有七种权限

resource：可以让用户在自己的表空间建表

dba：是一个最高权限的用户。

希望某个用户可以创建表

如果某个用户只有connect角色是不具有建表的能力的，所以要给他resource角色。

【案例】grant resource to 用户名

【desc 表名】查询表的结构。

希望某个用户可以去查询指定的表，不允许修改。

grant select on emp to 用户名

【案例】grant select on emp to xiaoming; 【这是scott用户给小明的权限】

select \* from scott.emp; 【这是xiaoming查询scott能让他查的表】

希望某用户可以去修改scott的emp表

【案例】grant update on emp to xiaoming 【Scott 给xiaoming赋权】

希望xiaoming用户可以去修改、删除、查询、添加scott的emp表

grant all on emp to xiaoming 【scott 赋权】

希望收回xiaoming对emp表的增删查改的权限

revoke all on emp from xiaoming;【撤回权限必须是那个用户给的权限那个用户撤销】

【对权限的维护】

希望xiaoming用户可以去查询scott的emp表/还希望小明可以把这个权限继续给别人？

【案例】grant all on emp to xiaoming with grant option【scott将权限给小明】

grant all on scott.emp to xiaohong;【xiaoming将权限给xiaohong】

【注意】 如果是对象权限，就加入with grant option。

如果是系统权限，system给xiaoming权限时：grant connect to xiaoming with admin potion。

？如果scott把xiaoming对emp表的查询权限回收，那么xiaohong会怎样

xiaohong的权限同样会被撤回。

1. 使用profile管理用户口令

【概述】profile是口令限制，资源限制的命令集合，当建立数据库时，Oracle会自动建立名称为default的profile。当建立用户没有指定profile选项，那么Oracle就会将default分配给用户。

1. 账户锁定

【概述】指定该用户（账户）登录时最多可以输入密码的次数，也可以指定用户锁定的时间（天）一般用dba的身份去执行该命令。

【案例】指定tea这个用户最多只能尝试3次登录，锁定时间为2天，让我们看看怎么实现？

sql>create profile lock\_account limit failed\_login\_attempts 3 password\_lock\_time 2;【创建profile文件】

sql>alter user tea profile lock\_account;【将profile限制分配给tea用户】

1. 给账户（用户）解锁

sql>alter user tea account unlock;

1. 终止口令

【概述】为了让用户定期修改密码可以使用终止口令的指令来完成，同样这个命令也需要dba身份来操作。

【案例】给前面创建的用户tea创建一个profile文件，要求该用户每隔10天修改自家的登录密码，宽限期为2天。

sql>create profile myprofile limit password\_life\_time 10 passsword\_grace\_time 2;

sql>alter user tea profile myprofile;

1. 口令历史

【概述】如果希望用户在修改密码时，不能使用以前使用过的密码，可以使用口令历史，这样Oracle就会将口令修改的信息存放到数据字典中，这样当用户修改密码时，Oracle就会对新旧密码进行比较，当发现新旧密码一样时，就提示用户重新输入密码。

【案例】

1. 建立profile。

sql>create profile password\_history limit password\_life\_time 10 password\_grace\_time 2 password\_reuse\_time 10

【注释】password\_reuse\_time：指定口令可重用时间即10天后就可以重用。

1. 分配给某个用户。
2. 删除profile

【概述】当不需要某个profile文件时，可以删除该文件。

【案例】sql>drop profile password\_history 【cascade】

***【练习题】***

1. deline 命令可以用于定义哪种变量？

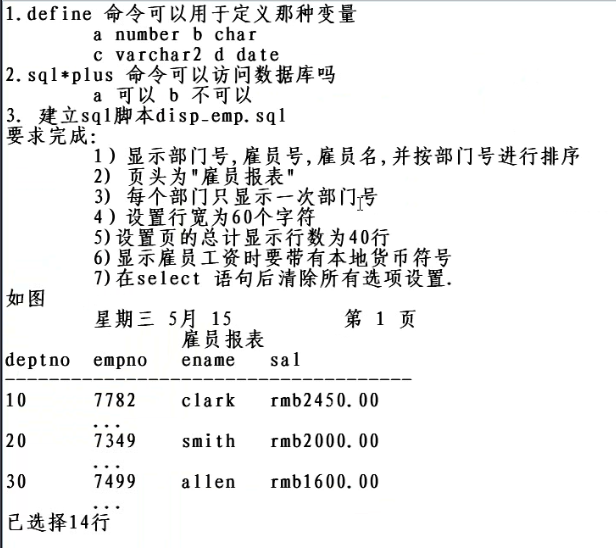
a number b char

c varchar2 d date

1. sql\*plus 命令可以访问数据库吗？

a 可以 b 不可以

1. 建立sql脚本 disp\_emp.sql，要求完成：
2. 显示部门号、雇员号、雇员名，并按部门号进行排序。
3. 页头为“雇员报表”。
4. 每个部门只显示一次部门号。
5. 设置行宽为60个字符。
6. 设置页的总计显示行数为40行。
7. 显示雇员工资时要带有本地货币符号。
8. 在select语句后清除所有选项设置。



1. 用户创建练习
2. 创建用户tea，stu，并给这两个用户resource，connect权限。
3. 使用scott用户把对emp表的select权限给tea
4. 使用tea查询scott的emp表
5. 使用scott用户把对emp表的所有权限赋给tea。
6. 使用tea更新/删除/插入scott的emp表。
7. 使用scott收回权限。

***第二部分 表的管理***

**【内容介绍】**

1. Oracle的表的管理
2. 基本查询
3. 复杂查询
4. Oracle数据库的创建

【期望目标】

1. 掌握Oracle表的管理（创建\维护）
2. 掌握对Oracle表的各种查询技巧
3. 学会创建新的Oracle数据库。

**【Oracle表的管理】**

1. 表名和列名的命名规则

* 必须以字母开头
* 长度不能超过30字符
* 不能使用Oracle的保留字
* 只能使用如下字符 A-Z ，a-z ，0-9 ，$ ，#等

1. Oracle支持的数据类型

**【字符型】**

1. char 定长 最大2000字符

【案例】char(10)’老于’前四个字符放’老于’，后添6个空格补全。

1. varchar2(20) 变长 最大4000字符。

【案例】varchar2(10)‘老于’ Oracle分配四个字符。这样可以节省空间

1. clob(character large object) 字符型大对象 最大4G

**【数字型】**

1. number 范围 -10的38次方~10的38次方

可以表示整数，也可以表示小数

1. number(5,2) 范围 -999.99~999.99

表示一个小数有5位有效数，2位小数

1. number(5) 范围 -99999~99999

表示一个五位整数

**【日期类型】**

1、date 包含年月日时分秒

2、timestamp 这是Oracle9i对date数据类型的扩展

**【图片类型】**

blob 二进制数据 可以存放图片/声音/视频 4G

*请说明char 和varchar的区别*

*1 、*

*char(n):定长字符串，该列中的数据长度一定是n如果长度小于n则会用空格补齐*

*varchar(n):表示变长字符串,该类中的数据最大长度为n,如果长度小于n则不会用空格补齐*

*2 、*

*char 最多存2000个字节*

*varchar最多存4000个字节*

*3 、*

*当以char类型字段作为查询条件效率要高于varchar类型字段查询*

1. 怎样创建表

【学生表】

sql>create table student(--表名

xh number(4), --学号*--指明列名和列的类型,数字4表示该表中该列能存储数字的个数*

xm varchar2(20), --姓名

sex char(2), --性别

birthday date，--出生日期

sal number(7,2) --奖学金 *--如果()中只写一个数字，表示整数和该整数的位数，如果是两个数，表示为浮点数，前边表示整数位数，后边表示小数位数*

);

*（）中的数字表示该表中该列能存储的字节个数，一个字母和数字占用一个字节，一个汉字占用两个字节*

【班级表】

sql>create table class(

classid number(2),

cname varchar2(20)

);

1. 修改表

【添加一个字段】

sql>alter table student add classid number(2);

【修改字段的长度】

sql>alter table student modify xm varchar2(30);

【修改字段的类型】*(如果该列已经有数据，那么修改的类型要可以兼容该数据)*

SQL>alter table student modify xm char(30)

【修改列的名字】前提不能有数据

SQL>alter table student rename column xm to sname

【删除一个字段】

sql>alter table student drop column sal;

【修改表的名字】

sql>rename student to stu;

【删除表】

sql>drop table student;

【给表添加注释】

SQL>comment on table table\_student is'学生信息表'

【给字段添加注释】

SQL>comment on column table\_student.sname is'学生姓名'

1. 添加数据

【所有字段都插入】

sql>insert into student values(‘A001’,’张三’,’01-5月-05’,10);

Oracle中默认的日期格式’DD-MON-YY’ dd日子（天） mon 月份 yy 2位的年 ‘09-6月-99’ 1999年6月9日

改日期的默认格式：alter session set nls\_date\_format = ‘yyyy-mm-dd’;

修改后，可以用我们熟悉的格式添加日期类型：insert into student value(‘A002’,’MIKE’,’男’,’1905-05-06’,10);

【插入部分字段】

sql>insert into student(xh,xm,sex) values(‘A003’,’JOHN’,’女’); *--对没有被赋值列默认null*

【插入空值】

sql>insert into student(xh,xm,se,birthday) values(‘A003’,’JOHN’,’女’,null);

？如果我们查询生日为空的学生

sql>select \* from student where birthday is null;

？如果我们查询生日不为空的学生

sql>select \* from student where birthday is not null;

【改一个字段】

sql>update student set sex=’女’ where xh=’A001’;

？将性别为男的sal减到原来的一般

sql>update student set sal=sal/2 where sex=’男’；

【修改多个字段】

sql>update student set sex=’男’,birthday=’1980-04-01’ where xh=’A001’;

【修改含有null值的数据】

不要使用=null，，，要使用is null 来进行查询

1. 删除数据

【删除所有记录】

①delete from student;

删除所有记录，表结构还在，写日志，可以恢复的，速度慢。使用savepoint name; 和rollback;命令完成回滚。

案例：sql>save point aa;--设置保存点，将某时刻保存日志，如果在后边做了误操作，我们可以返回到这个点。使从这个时间点之后的操作全部回滚。

--保存点在不作处理的情况下，默认只有一个。

sql>delete from student ;

sql>rollback aa;--回滚到保存点aa。

②truncate table student;删除表中的所有记录，表结构还在，不写日志，无法找回删除的记录，速度快。

【删除表的结构和数据】

drop table student;

【删除一条记录】

delete from student where xh=’A001’;

1. 创建序列sequence

create sequence seq\_dept *--创建一个序列，名为seq\_dept*

start with 80 *--设定序列的起始值*

minvalue 80 *--设定序列的最小值*

maxvalue 99 *--设定序列的最大值*

increment by 1 *--每次序列的增加值*

nocache *--是否缓存，以及缓存数*

nocycle *--是否循环*

***查询表是对表操作的最重要的操作***

1. 表的基本查询--简单的查询语句

【介绍】

在我们讲解的过程中我们利用scott用户存在的几张表（emp，dept）为大家演示如何使用select语句，select语句在软件编程中非常的有用，希望大家好好的掌握。

clear;这是一个清屏命令

【查看表的结构】

sql>desc dept;

【查询所有列】

sql>select \* from dept;

【查询指定列】

sql>select ename,sal,job,deptno from emp;

【如何取消重复行】

sql>select distinct deptno,job from emp;

【比较用\*号查询和用字段查询的速度】

创建一个表，使用疯狂复制法进行赋值

sql>create table users(userid varchar2(20),username varchar2(30),userpass varchar2(30));

sql>insert into users(‘a001’,’小明’,’bruceyu’);

sql>insert into users(userid,username,userpass) select \* from users;

sql>insert into users(userid,username,userpass) select \* from users;

sql>insert into users(userid,username,userpass) select \* from users;

sql>insert into users(userid,username,userpass) select \* from users;

sql>insert into users(userid,username,userpass) select \* from users;

......

sql>select count(\*) from users;---查看表中有多少条数据

查看两种查询方式的查询时间

sql>set timing on;--打开操作时间

sql>select ename from emp;

sql>relect \* from emp;

由此可见，不要轻易使用select \* from 语句

？查询SMITH的薪水，工作，所在部门。

sql>select deptno,job,sal from emp where ename=‘SMITH’;--Oracle中对命令、字段不区分大小写，但是内容区分大小写。

【使用算数表达式查询】

？显示每个雇员的年工资

SQL> select ename,(sal+nvl(comm,0))\*12 as yearsal from emp;

【使用列的别名】*可以通过别名改变查询结果的表头*

select ename ‘姓名’ , sal\*12 as ‘年收入’ from emp;--对于类别可以使用as 也可以不适用，但是对于表别名一定没有。

sql>select ename ‘姓名’，sal\*12 as ‘年收入’ from emp;

--但如果加上奖金

sql>select ename ‘姓名’，sal\*12+comm\*12 as ‘年收入’ from emp;--这样会出现奖金为null的人的年收入是空的。？？？？

↓

【如何处理null值】

使用nvl函数来处理

sql>select ename ‘姓名’，sal\*12+nvl(comm,0)\*12 as ‘年收入’ from emp;

【如何连接字符串（||）】

select ename || ‘is a’ || job from emp;

【条件查询】

*在Oracle中使用case when then 进行条件判断*

*--显示员工姓名，工资，以及工资等级大于5000为A [3000,5000] 为 B[1500,3000] C小于1500为D*

select ename,

sal,

case

when sal >5000then

'A'

when sal >3000then

'B'

when sal >1500then

'C'

else

'D'

endas 等级

from emp

*--统计 员工中 工资大于5000的人数 [3000,5000]的人数,[1500,3000]的人数,1500以下的人数*

select

case

when sal >5000then

'A'

when sal >3000then

'B'

when sal >1500then

'C'

else

'D'

endas 等级,count(\*)as 人数

from emp

groupby

case

when sal >5000then

'A'

when sal >3000then

'B'

when sal >1500then

'C'

else

'D'

end

orderby 等级 asc

【使用where子句】*如果在需要在查询中加入条件(仅仅显示满足条件的信息)使用where 子句。注意 where 后一定要返回布尔类型*

？如何显示工资高于3000的员工

sql>select ename,sal from emp where sal>3000;

？如何查找1982.1.1后入职的员工

sql>select ename,hiredate from emp where hiredate>’1-1月-1982’;

？如何显示工资在2000到2500的员工情况

sql>select enme,sal from emp where sal>=2000 and sal<=2500;

【使用like操作符】

如果对于字符串类型字段的内容进行部分匹配，则需要模糊查询。

注意比较符 为 like +通配符

模糊查询中通配符就两种:

%：表示任意的0到多个字符

\_：表示任意的单个字符

？如何显示首字母为S的员工的姓名和工资

sql>select ename,sal from emp where ename like ‘S%’;

？如何显示第三个字符为大写O的所有员工的姓名和工资

sql>select ename,sal from emp where ename like ‘\_\_O%’;

【在where条件中使用in】

？如何显示empno为123,345,800......的雇员情况

sql>select \* from emp where empno in(123,345,800....);

【使用is null的操作符】

？如何显示没有上级的雇员的情况

sql>select \* from emp where mgr is null;

【使用逻辑操作符号】

？查询工资高于500或是岗位为MANAGER的雇员，同时还要满足他们的姓名首字母为大写的J

sql>select \* from emp where (sal>500 or job=’MANAGER’) and ename like ‘J%’;

【使用order by 语句】

？如何按照工资的从低到高的顺序显示雇员的信息

sql>select \* from emp order by sal ;--默认情况下是升序asc可以省略，降序为 desc。

？按照部门号升序而雇员的工资降序排列

sql>select \* from emp order by deotno asc , sal desc;

【使用列的别名排序】

sql>select ename,sal\*12 “年薪” from emp order by “年薪” asc;

列名需要使用“”圈中

【分页查询】

按雇员的id号升序取出

**~~学完子查询后再探讨~~**

1. Oracle表的复杂查询

【说明】

在实际应用中经常需要执行复杂的数据统计，经常需要显示多张表的数据，现在我们给大家介绍较为复杂的select语句。

【数据分组—max，min, avg, sum, count】*又称为聚合函数。用于对某一列进行统计运算。*

*注意,聚合函数单独使用一定是返回一行一列*

*--max(列):返回指定列中最大的值*

*--查询出公司中的最高工资*

selectmax(sal)from emp;*--注意,聚合函数单独使用一定是返回一行一列*

*--min(列):返回指定列中最小的值*

selectmin(sal)from emp

*--sum(数字列)：对指定列求和*

selectsum(sal)from emp;*--计算公司的总月工资*

*--avg(数字列):得到指定列的平均值*

selectavg(sal)from emp

*--count(\*):得到结果集的总条目数(计算结果集一共有多少条记录)*

selectcount(\*)from emp

*--注意：null不参与聚合运算，聚合函数会自动忽略null值*

Select \* from emp

Select count(comm)from emp

Select avg(nvl(comm,0))from emp

Select count(\*)from emp where sal>60000

？如何显示所有员工中最高工资和最低工资

SQL>select max(sal),min(sal) from emp;

？查出最高工资的员工的名字和工资

SQL>select ename,sal from emp where sal=(select max(sal) from emp);

？显示所有员工的平均工资和工资总和

SQL> select avg(sal),sum(sal) from emp;

？计算共有多少员工

SQL> select count(\*) empnum from emp;

扩展要求：

？请显示工资最高的员工的名字，工作岗位

SQL> select ename,job from emp where sal=(select max(sal) from emp);

？请显示工资高于平均工资的员工信息。

SQL> select \* from emp where sal>(select avg(sal) from emp);

【多表联合】

*如果在查询过程中，所需要的数据来自于多张表，则需要将这些表联合起来进行查询*

*select ... from 表 join 表2 on 表的连接条件*

*多表联合分为:*

*1、内联([inner] join):仅仅显示满足连接条件信息的数据*

*2、外联(left|right [outer] join):在显示满足连接条件信息的数据的基础上,*

*还有将左(右)侧信息都显示出来(着重显示),对侧表的数据用null不起*

?查询工资大于2000的 员工姓名,工资,职位,和所在部门的名称

SQL>**select ename,sal,job,dname from emp join dept on emp.deptno = dept.deptno where sal>2000**

?查询员工的姓名，工资和工资等级

SQL>**select emp.\*,grade from emp join salgrade on sal>=losal and sal<=hisal**

?查询员工姓名,工资,职位，所属部门名称,工资等级

SQL>**select ename,sal,job,dname,grade from emp join dept on emp.deptno = dept.deptno join salgrade on sal>=losal and sal<=hisal**

？显示每个员工的姓名,工资和所属部门

SQL>select ename,sal,dname from emp innerjoin dept on emp.deptno = dept.deptno

SQL>select ename,sal,nvl(dname,'部门未定')as 部门名 from dept rightjoin emp on emp.deptno = dept.deptno

【group by 和 having子句】

group by用于对查询的结果分组统计，

having子句用于限制分组显示结果。

*分组查询利用group by 对结果集根据指定的列进行分组。*

*分组查询的基本思路:对结果集(查询好的信息)根据指定的列进行分组，并对每个组进行聚合运算*

*通常分组查询与聚合函数联合使用*

*即:在分组查询中,select后只能加分组列和聚合函数(在这里聚合函数是对每组进行运算)*

*(在查询中聚合函数如果要和普通列一起显示，那么这个列必须是分组列)*

*当对聚合结果进行过滤时使用having*

？如何显示每个部门的平均工资和最高工资

SQL> select avg(sal),max(sal),deptno from emp group by deptno;

？显示每个部门的每种岗位的平均工资和最低工资

SQL> select min(sal),avg(sal),max(sal),deptno,job from emp group by deptno,job;

？显示平均工资低于2000的部门号和它的平均工资

SQL> select deptno,avg(sal) from emp group by deptno having avg(sal)<2000;

扩展要求：

【对数据分组的总结】

1. 分组函数只能出现在选择列表、having、order by 子句中
2. 如果在select 语句中同时包含有group by , having , order by 那么他们的顺序是group by , having , order by
3. 在选择列中如果有列、表达式和分组函数，那么这些列和表达式必须有一个出现在group by 子句中，否则就会出错

SQL> select deptno,avg(sal) from emp group by deptno having avg(sal)>2000 order by avg(sal)desc;

如select deptno,avg(sal),max(sal) from emp group by deptno having avg(sal)<2000;这里的deptno就一定要出现在group by 中.

1. Oracle表的多表查询

【说明】

多表查询是指基于两个和两个以上的表或是视图的查询。在实际应用中，查询单个表可能不能满足你的需求，(显示sales部门位置和其员工的姓名)，这种情况下需要使用到(dept表和emp表)

？显示雇员,雇员工资及所在部门的名字【笛卡尔集】

规定：多表查询的条件是 至少不能少于表的个数-1

SQL> select a1.ename ,a1.sal, a2.dname from emp a1,dept a2 where a1.deptno=a2.deptno;

？如何显示部门号为10的部门名、员工名和工资

SQL> select a1.dname,a2.ename,a2.sal from dept a1,emp a2 where a1.deptno=a2.deptno and a1.deptno=10;

？显示各个员工的姓名，工资，及其工资的级别*--可以使用 字段 between 值1 and 值2*

SQL> select a1.ename,a1.sal,a2.grade from emp a1,salgrade a2 where a1.sal between a2.losal and a2.hisal;

扩展要求：

？显示雇员名，雇员工资及所在部门的名字，并按部门排序。

SQL> select a1.ename,a2.dname,a1.sal from emp a1,dept a2 where a1.deptno=a2.deptno order by a1.deptno;

【自连接】

自连接是指在同一张表的连接查询

？显示某个员工的上级领导的姓名（比如显示“FORD”的上级）

SQL> select worker.ename,boss.ename from emp worker,emp boss where worker.mgr=boss.empno and worker.ename='FORD';

【子查询】

什么是子查询：子查询是指嵌入在其它sql语句中的select语句，也叫嵌套查询。

*在Sql语句中有的数据需要由另一个查询来提供。那么这个提供数据的查询就称为子查询*

*有时需要子查询提供一张临时表(即将查询结果看成一张表)。注意必须保证所有的计算列，必须起别名*

【单行子查询】

单行子查询是指返回一行数据的子查询语句

？请思考：如何显示与SMITH同一部门的所有员工

SQL> select ename from emp where deptno=(select deptno from emp where ename='SMITH');

\*\*数据库执行sql是从左到右扫描的。

【多行子查询】

多行子查询指返回多行数据的子查询

？请思考：如何查询和部门10的工作相同的雇员的名字、岗位、工资、部门号。

SQL> select \* from emp where job in(select distinct job from emp where deptno=10);

【在多行子查询中使用all操作符】

？请思考：如何显示工资比部门30的所有员工的工资高的员工的姓名、工资和部门号

SQL> select ename,sal,deptno from emp where sal>all(select sal from emp where deptno=30);

扩展要求：

想一想还有没有别的查询方法。

可以使用分组函数找出30号部门的最高工资。

SQL> select ename,sal,deptno from emp where sal>(select max(sal) from emp where deptno=30);【执行效率比all的高】

【在多行子查询中使用any操作符】

?请思考:如何显示工资比部门30的任意一个员工的工资高的员工的姓名、工资和部门号

SQL> select ename,sal,deptno from emp where sal>any(select sal from emp where deptno=30);

扩展要求：

想一想还有没有别的查询方法

SQL> select ename,sal,deptno from emp where sal>(select min(sal) from emp where deptno=30);

【多列子查询】

单行子查询是指子查询只返回单列、单行数据，多行子查询是指返回单列多行数据，都是针对单列而言的，而多列子查询则是只查询返回多个列数据的子查询语句

？请思考：如何查询与SMITH的部门和岗位完全相同的所有雇员

SQL> select \* from emp where (deptno,job)=(select deptno,job from emp where ename='SMITH');

【在from子句中使用子查询】

？请思考：如何显示高于自己部门平均工资的员工的信息

SQL> select a2.ename,a2.sal,a2.deptno,a1.mysal from emp a2,(select deptno,avg(sal) mysal from emp group by deptno) a1 where a2.deptno=a1.deptno and a2.sal>a1.mysal;

这里需要说明的是：当在from子句中使用子查询时，该子查询会被作为一个视图来对待，因此也叫作内嵌视图，当在from子句中使用子查询时，必须给子查询指定别名。给表起别名不能加as。列的别名可加可不加。

【Oracle分页查询】

按雇员的ID号升序取出。

Oracle分页一共有三种方式

1. 按rownum分页[这种方式最慢]

SQL> select \* from emp;

显示rownum[Oracle分配的]

[Oracle中的rownum在where语句中只能使用一次,使用and 语句选择范围时是查询不出来的]

SQL> select \* from (select a1.\*,rownum rn from (select \* from emp) a1 where rownum<=10) where rn>=6;

红色部分是固定写法不能改动

1. 按rowid分页

SQL>select \* from emp where rowid in(select rid from (select rownum rn,rid from(select rowid rid,empno from emp order by empno desc) where

rownum<6) where rn>3) order by empno desc;

其中emp为表名称,empno为表关键字段,取按empno降序排序后的4~5条记录。

执行速度最快,但是相对难理解。

1. 按分析函数来分

SQL>select \* from (select t.\*, row\_number() over(order by empno desc) rk from emp t) where rk < 10 and rk > 5;

1. 几个查询变化
   1. 指定查询列,只需要修改最里层的子查询
   2. 如何排序，只需修改最里层的子查询。
   3. 指定要显示的条目，只需要改动那两个数即可。

【用查询结果创建新表】--表的复制

这个命令是一种快捷的建表方法。

SQL> create table myemp(id,name,sal,job,deptno) as select empno,ename,sal,job,deptno from emp;

1. 合并查询

有时在实际应用中，为了合并多个select语句的结果，可以使用集合操作符号 union，union all，intersect，minus。

1. union【并集】

该操作符用于取得两个结果集的并集。当使用该操作符时，会自动去掉结果集中重复行。

SQL> select ename,sal,job from emp where sal>2500 union select ename,sal,job from emp where job='MANAGER';

1. union all【合集】

该操作符与union相似，用于取得两个结果集的并集。当使用该操作符时，不会自动取消重复行，而且不会排序

SQL> select ename,sal,job from emp where sal>2500 union all select ename,sal,job from emp where job='MANAGER';

1. intersect【交集】

使用该操作符用于取得两个结果集的交集。

SQL> select ename,sal,job from emp where sal>2500 intersect select ename,sal,job from emp where job='MANAGER';

1. minus【差集】

使用该操作符用于取得两个结果集的差集，他只会显示存在第一个集合中，而不存在第二个集合中的数据。

SQL> select ename,sal,job from emp where sal>2500 minus select ename,sal,job from emp where job='MANAGER';

1. 创建新的数据库

【创建数据库有两种方法】

1. 通过Oracle提供的向导工具。√

DBCA[数据库配置助手]

1. 我们可以用手工步骤直接创建。

这种方式在后边讲解.

***第三部分 Oracle操作数据和事务处理***

【内容介绍】

1. Java程序如何操作Oracle
2. 如何在Oracle中操作数据
3. Oracle事务处理
4. sql函数的使用

【期望目标】

1. 掌握Oracle表对数据库操作技巧
2. 掌握在Java程序中操作Oracle
3. 理解Oracle事务概念
4. 掌握Oracle各种sql函数

【Java连接Oracle】

【介绍】

前面我们一直在plsql中操作Oracle,那么如何在Java程序中操作数据库呢?

下面我们举例说明,写一个ShowEmp.java,分页显示emp表的用户信息。

【在Oracle中操作数据】-使用特定格式插入日期值

【使用to\_date函数】

请大家思考:如何插入列带有日期的表,并按照年—月—日的格式插入？

SQL> insert into emp(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values(8888,'BRUCE','PRESIDENT',7788,to\_date('2017-1-20','yyyy-mm-dd'),10000,2000,20);

*--找到在1981-5-1到 1982-2-1之间入职的员工*

select\*from emp where hiredate between to\_date('1981-05-01','yyyy-mm-dd')and to\_date('1982-02-01','yyyy-mm-dd')

*--查找10部门和20部门中工资大于2000的员工（默认 and优先于 or）*

select\*from emp where(deptno =10or deptno=20)and sal>2000

*--查询所有在4月份入职的员工*

select\*from emp where to\_char(hiredate,'mm')='04'

【在Java中操作数据】-使用子查询插入数据

【介绍】

当使用values子句时，一次只能插入一行数据，当使用子查询插入数据时，一条insert语句可以插入大量的数据。当处理行迁移或者装载外部表的数据到数据库时，可以使用子查询来插入数据。

？将emp表中10号部门的数据导入到mytable表中

SQL> create table mytable(myid number(4),myname varchar2(50),mydept number(5));

SQL> insert into mytable(myid,myname,mydept) select empno,ename,deptno from emp where deptno=10;

SQL> select \* from mytable;

【Oracle中的事务处理】

【介绍】

使用update语句更新数据时，既可以使用表达式或者数值直接修改数据，也可以使用子查询修改数据。

？希望员工scott的岗位、工资、补助与SMITH员工一样

【事务】

什么是事务：事务用于保证数据的一致性，它由一组相关的dml语句组成,该组的dml语句要么全部成功,要么全部失败.

如:网上转账就是典型的要用事务来处理,用以保证数据的一致性.

【事务和锁】

当执行事务操作时（dml语句），Oracle会在被作用的表上加锁，防止其他用户改表的结构。这里对我们用户来讲是非常重要的。（类似于排队上厕所）

【提交事务】

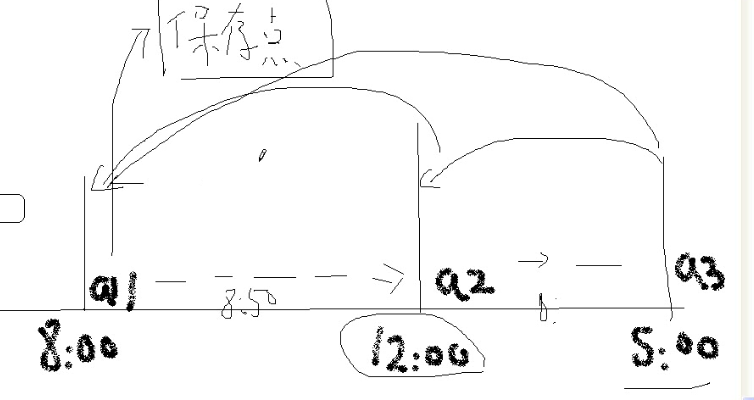
当执行使用commit语句可以提交事务。当执行了commit语句之后，会确认事务的变化、结束事务、删除保存点、释放锁，当使用commit语句结束事务之后，其他会话将可以查看到室外变化后的新数据。

【回退事务】

在介绍回退事务前，我们先介绍一下保存点（savapoint）的概念和作用.保存点是事务中的一点.用于取消部分事务,当结束事务时,会自动的删除该事务所定义的所有保存点.当执行rollback时,通过制定保存点可以回退到指定的点这里.我们作图说明.

SQL> savepoint a1;

SQL> rollback to a1;



【事务的几个重要操作】

1. 设置保存点

SQL> savepoint a;

1. 取消部分事务

SQL> rollback to a;

1. 取消全部事务

SQL> rollback;

【Java程序中如何使用事务】

在Java操作数据库时，为了保证数据一致性，比如转账操作：

1. 从一个账户减掉10$
2. 在另一个账户上加入10$,我们看看如何使用事务?

【只读事务】

只读事务是指只允许执行查询的操作，而不允许执行任何其他dml操作的事务，使用只读事务可以确保用户只能取得某时间点的数据。假定机票代售点每天18点开始统计今天的销售情况，这时可以使用只读事务。在设置了只读事务后，尽管其他会话可能会提交新的事务，但是只读事务将不会取得最新数据的变化，从而可以保证取得特定时间点的数据信息。

【设置只读事务】

SQL> set transaction read only;

**【SQL函数的使用】**

【字符函数】

【介绍】

字符函数是Oracle中最常用的函数，我们来看看有哪些字符函数：

1. lower（char）:将字符串转化为小写的格式
2. upper(char):将字符串转化为大写的格式
3. length(char):返回字符串的长度
4. substr(char,m,n):去字符串的子串.--表示从char字符串中的第m位开始截取n个字符.

？将所有员工的名字按小写的方式显示

SQL> select lower(ename) as ename from emp;

？将所有员工的名字按大写的方式显示

SQL> select upper(ename) as ename from emp;

？显示正好为5个字符的员工的姓名。

SQL> select \* from emp where length(ename)=5;

？显示所有员工姓名的前三个字符。

SQL> select substr(ename,1,3) from emp;

？以首字母大写的方式显示所有员工的姓名。

SQL> select upper(substr(ename,1,1))||lower(substr(ename,2,length(ename)-1)) as ename from emp;

？以首字母小写的方式显示所有员工的姓名。

SQL> select lower(substr(ename,1,1))||upper(substr(ename,2,length(ename)-1)) as ename from emp;

【介绍】

字符函数是Oracle中最常用的函数，我们来看看有哪些字符函数：

1. replace（char1,search\_string,replace\_string）:字符串替换
2. instr(char,string,n,m)取子串在字符串中的位置

char表示表中的字段

string表示所有查询的字符串

n表示从第几位开始查

m表示到第几次出现

*instr(string, search\_string, startpos, occurrence)--在字符串string中startpos位置向后查找search\_string，第occurrence次出现的位置，如果没有返回0*

?查找第5个字母为E的员工信息

**SQL>select \* from emp where instr(ename,'E',5,1)=5;*--indexOf***

？显示所有员工的姓名，用a替换所有的”A”

**SQL> select replace(ename,'A','a') from emp;**

？查出员工姓名中只含有2个A的员工信息

SQL> select \* from emp join (select empno, length(emp.ename) - length(replace(emp.ename, 'A', '')) as nums from emp) t on emp.empno = t.empno and nums = 2;

SQL> select \* from emp where instr(ename,'A',1,2)>0 and instr(ename,'A',1,3)=0;

！该问题也可以使用函数来解决

SQL> select \* from emp join (select empno, lengthb(regexp\_replace(emp.ename,'[^A]',null)) as nums from emp) t on emp.empno = t.empno and nums = 2;

【数学函数】

【介绍】

数学函数的输入参数和返回值的数据类型都是数字类型的，数学函数包括cos,cosh,exp,ln,log,sin,sinh,sqrt,tan,tanh,acos,asin,atan,round,我们将最常用的:

1. round（n,[m]）
2. trunc(n,[m])
3. mod(m,n)
4. floor(n)
5. ceil(n)

【介绍】

* round(n,[m])该函数用于执行四舍五入,如果省略掉m,则四舍五入到整数;如果m是正数,则四舍五入到小数点的m位后.如果m是负数,则四舍五入到小数点的m位前

SQL> select round(sal) from emp;

SQL> select sal,round(sal,1) from emp;

* trunc(n,[m]) 该函数用于截取数字.如果省掉m,就截取小数部分,如果m是正数就截取到小数点的m位后,如果m是负数,则截取到小数点的前m位

SQL> select trunc(comm,1),comm from emp ;

* mod(m,n) 取模
* floor(n) 返回小于或是等于n的最大整数(向下取整)
* ceil(n) 返回大于或是等于n的最小正数(向上取整)

案例数据:2345.34 45.94

？显示在一个月为30天的情况下所有员工的日薪金，忽略余数。

SQL> select trunc(sal/30),ename from emp;

SQL> select floor(sal/30),ename from emp;

对数字的处理,在财务系统或银行系统中用的最多,不同的处理方法,对财务报表有不同的结果.

\*\*在做Oracle测试时，我们使用亚元表dual.

*dual是oracle中的虚表，当在查询中仅仅进行运算，或者函数，或者查看系统(公共)字段使用dual保证了语法完整*

*--sysdate 是oracle中的一个系统字段，用于记录数据服务器当前时间*

select sysdate from dual;

*--在Oracle中可以通过 sys\_guid() 来提供一个32位不重复码*

selectlength(sys\_guid()),sys\_guid()from dual

【介绍】

其他的数学函数：

1. abs(n):返回数字n的绝对值

SQL> select abs(-13) from dual;

1. acos(n) 返回数字的反余弦值
2. asin(n) 返回数字的反正弦值
3. atan(n) 返回数字的反正切
4. cos(n)
5. exp(n) 返回e的n次幂
6. log(m,n) 返回对数值
7. power(m,n) 返回m的n次幂

【日期函数】

【介绍】

日期函数用于处理date类型的数据。

默认情况下日期格式是dd-mon-yy 即12-7月-17

1. sysdate:该函数返回系统时间

SQL> select sysdate from dual;

1. add\_months(d,n)
2. last\_day(d):返回指定日期所在月份的最后一天

？查找已经入职400个月多的员工

SQL> select \* from emp where sysdate>add\_months(hiredate,400);

？显示满10年服务年限的员工的姓名和受雇日期。

SQL> select \* from emp where sysdate>=add\_months(hiredate,12\*10);

？对于每个员工，显示其加入公司的天数。

SQL> select ename,floor(sysdate-hiredate) as "入职天数" from emp;

？找出个月倒数第三天受雇的所有员工。

SQL> select hiredate,ename from emp where last\_day(hiredate)-2=hiredate;

【转换函数】

【介绍】√

转换函数用于将数据类型从一种转为另外一种，在某种情况下，Oracle server允许值的数据类型和实际的不一样，这时Oracle server会隐含的转化数据类型。

比如：

create table t1(id number);

insert into t1 values(‘10’) ;-->这样Oracle会自动的将’10’→10

create table t2(id varchar2(10));

insert into t2 values(1);-->这样Oracle就会自动的将1→’1’;

我们要说的是尽管Oracle可以进行隐含的数据类型的转换,但是他并不适应于所有情况,为了提高程序的可靠性,我们应该使用转换函数进行转换.

【to\_char】

你可以使用select 使用 select ename,hiredate,sal from emp where deptno = 10; 显示信息,可是,在某些情况下,这个不能满足你的需求。

|  |  |
| --- | --- |
| 时间显示符 | 货币显示符 |
| yy:两位数字的年份:2017-->17 | 9:显示数字,并忽略前面的0 |
| yyyy:四位数字的年份 2017 | 0:显示数字,如果位数不足,则用0补齐 |
| mm:两位数字的月份 8月-->08 | $ 美元符 |
| dd:两位数字的天 20号-->20 | L本地货币标识符 |
| hh24:晚上8时---->20 | C人民币标识符 |
| hh12:晚上8时--->08 | G |
| mi、ss:显示分钟\秒 | D |
|  | .:在指定位置显示小数点 |
|  | ,:在指定位置显示逗号 |

？日期是否可以显示时-分-秒

SQL> select ename,to\_char(hiredate,'yyyy-mm-dd hh24:mi:ss') from emp;

？薪水是否可以显示指定的货币符号

SQL> select to\_char(sal,'L99,999.99') from emp;

？显示1980年入职的所有员工

SQL> select \* from emp where to\_char(hiredate,'yyyy')=1980;

？显示所有12月份入职的员工

SQL> select \* from emp where to\_char(hiredate,'mm')=12;

【to\_date】

函数to\_char用于将字符串转换成date类型的数据.

？能否按照中国人习惯的方式年-月-日添加日期

SQL>select to\_char(sysdate,'yyyy')from dual

SQL>select to\_char(sysdate,'yyyy"年"mm"月"dd"日"')from dual;

【系统函数】

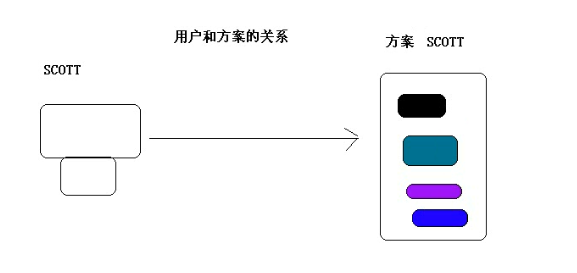
【sys\_context】

1. terminal：当前会话客户所对应的终端的标识符
2. language：语言
3. db\_name：当前数据库名称
4. nls\_date\_format：当前会话客户所对应的日期格式
5. session\_user：当前会话客户所对应的数据库用户名
6. current\_schema：当前会话用户所对应的默认方案名？
7. host：返回数据库所在主句的名称

通过该函数，可以查询一些重要信息，比如你正在使用哪个数据库？

SQL> select sys\_context('USERENV','db\_name') from dual;

红色部分固定不变，不能改，但不区分大小写。



【分析函数】

？显示员工信息和他的工资在本部门的排名

【row\_number()over()】

顺序排序，不回去统计是否并列

SQL> select emp.\*,row\_number()over(partition by deptno order by sal) row\_number from emp;

【rank()over()】

跳跃性排序（非密集排序），假设如果有两个2并列，那么第3位排名会是4，将会跳过3

SQL> select emp.\*,rank()over(partition by deptno order by sal) row\_number from emp;

【dense\_rank()over()】

非跳跃性排序（密集排名函数），假设如果有两个2并列，那么第3位排名是3，不会跳过3

SQL> select emp.\*,dense\_rank()over(partition by deptno order by sal) row\_number from emp;

***第四部分 数据库管理***

【内容介绍】

1. 数据库管理员
2. 数据库（表）的逻辑备份与恢复√
3. 数据字典和动态性能视图√
4. 管理表空间和数据文件 √

【期望目标】

1. 了解Oracle管理员的基本职责
2. 掌握备份和恢复数据库/表的方法
3. 理解表空间、数据字典、性能视图。

【数据库管理员】

【介绍】

每个Oracle数据库应该至少有一名数据库管理员（dba），对于一个小的数据库，一个dba就够了，但是对于一个大的数据库可能需要多个dba分别担负不同的管理职责。

那么一个数据库管理员的主要工作是什么呢？

【职责】

1. 安装和升级Oracle数据库
2. 建库，表空间，表，视图，索引……
3. 制定并实施备份与恢复计划
4. 数据库权限管理，调优，故障排除
5. 对于高级dba，要求能参与项目开发，会编写sql语句、存储过程、触发器、规则、约束、包。

【数据库管理员的用户主要是sys和system】--sys是董事长，--system是总经理

在前面我们已经提到这两个用户，区别主要是：

1. 最重要的区别，存储的数据重要性不同

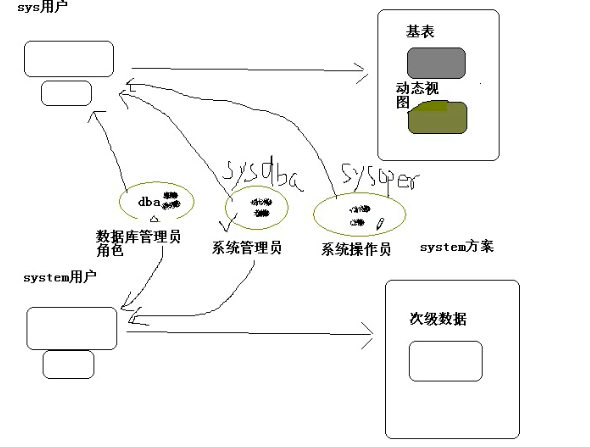
①sys：所有Oracle的数据字典的基表和视图都存放在sys用户中，这些基表和视图对于Oracle的运行是至关重要的，由数据库自己维护，任何用户都不能手动更改。sys用户拥有dba，sysdba，sysoper角色或权限,是Oracle权限最高的用户.

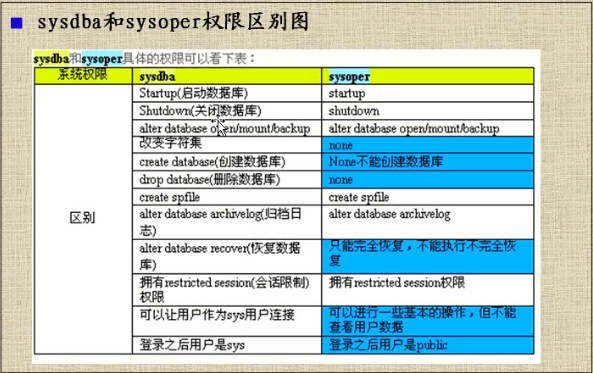
②system：用于存放次一级的内部数据，如Oracle的一些特性或工具的管理信息。system用户拥有dba，sysdba角色或系统权限。

1. 其次的区别，权限的不同

①sys用户必须以as sysdba或sysoper形式登录.不能以normal方式登录数据库。

②system如果正常登陆，他其实就是一个普通的dba用户，但是如果以as sysdba登录，其结果实际上它是作为sys用户登录的，从登录信息里面我们可以看出来。





【dba权限的用户】

dba用户是指具有dba角色的数据库用户。特权用户可以执行启动实例，关闭实例等特殊操作，而dba用户只有在启动数据库后才能执行各种管理工作。

sysdba>sysoper>dba

【管理初始化参数】

初始化参数用于设置实例或是数据库的特征。Oracle9i提供了200多个初始化参数，并且每个初始化参数都有 默认值。

1. 显示初始化参数

show parameter命令

1. 如何修改参数

需要说明的是如果你希望修改这些初始化的参数，可以到文件C:..\oracle\admin\ORCLYHY\pfile\init.ora文件中去修改不要修改实例的名字。

【数据库（表）的逻辑备份与恢复】

【介绍】

逻辑备份是指使用工具export将数据对象的结构和数据导出到文件的过程，逻辑恢复是指当数据库对象被误操作而损坏后使用工具import利用备份的文件把数据对象导入到数据库的过程。

物理备份即可在数据库open的状态下进行也可在关闭数据库后进行，但是逻辑备份和恢复只能在open的状态下进行。

【导出】

导出具体的分为：导出表，导出方案，导出数据库三种方式……

导出使用exp命令来完成的，该命令常用的选项有：

userid： 用于指定执行导出操作的用户名，口令，连接字符串。

tables： 用于指定执行导出操作的表。

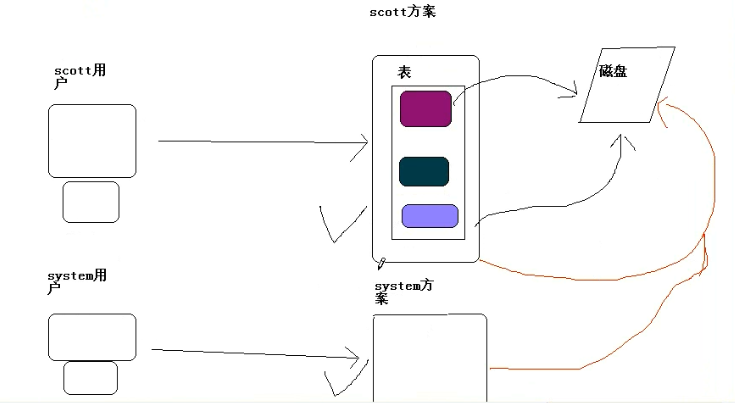
owner： 用于指定执行导出操作的方案。

full=y： 用于指定执行导出操作的数据库。

inctype：用于指定执行导出操作的增量类型。

rows： 用于指定执行导出操作是否要导出表中的数据。

file： 用于指定导出文件名。



【导出表】

【特别说明】

在导入导出的时候，要到Oracle目录的bin目录下.

1. 导出自己的表

exp userid=scott/tiger@ORCLYHY tables=(emp) file=d:\e1.dmp





1. 导出其他方案的表

如果用户要到处其他方案的表，则需要dba的权限或是exp\_full\_database的权限,比如system就可以导出scott的表

exp userid=system/tiger@ORCLYHY tables=(scott.emp) file=d:/e2.dmp



1. 导出表的结构

exp userid=scott/tiger@accp tables=(emp) file=d:\e1.dmp rows=n

1. 使用直接导出方式

exp userid=scott/tiger@accp tables=(emp) file=d:\e1.dmp direct=y

这种方式比默认的常规方式速度要快,当数据量大时,可以考虑使用这样的方法.

这时需要数据库的字符集要与客户端字符集完全一致,否则会报错......

【导出方案】

导出方案是指使用export工具导出一个方案或是多个方案中的所有对象（表，索引，约束……）和数据，并存放到文件中。

1. 导出自己的方案

exp userid=scott/tiger@ORCLYHY owner=scott file=d:\scott.dmp

1. 导出其他方案

如果用户要导出其他方案,则需要dba的权限或是exp\_full\_database权限,例如system用户就可以导出任何方案

exp userid=system/tiger@ORCLYHY owner=(system,scott) file=d:\system.dmp

【导出数据库】

导出数据库是指利用export导出所有数据库中的对象及数据。要求该用户具有dba的权限或是exp\_full\_database权限

exp userid=system/tiger@ORCLYHY full=y inctype=complete file=d:\x.dmp

因为数据量大,所以耗用的时间会较长.

【导入】

【介绍】

导入就是使用import将文件中的对象和数据导入到数据库中，但是导入要使用的文件必须是export所导出的文件。与导出相似，导入也分为导入表，导入方案，导入数据库三种方式。

imp常用的选项有

userid：用于指定执行导入操作的用户名，口令，连接字符串

tables：用于指定执行导入操作的表

fromuser：用于指定源用户

touser：用于指定目标用户

file：用于指定导入文件名

full=y：用于指定执行导入整个文件

inctype：用于指定执行导入操作的增量类型

rows：指定是否要导入表行（数据）

ignore：如果表存在，则只导入数据。

【导入表】

1. 导入自己表

imp userid=scott/tiger@ORCLYHY tables=(emp) file=d:\xx.dmp

1. 导入表到其他用户

要求该用户具有dba权限，或是imp\_full\_database

imp userid=system/tiger@ORCLYHY tables=（emp）file=d:\xx.dmp touser=scott

1. 导入表的结构

只导入表的结构而不导入数据

imp userid=scott/tiger@ORCLYHY tables=(emp) file=d:\xxx.dmp rows=n

1. 导入数据

如果对象(如比表)已经存在可以只导入表的数据

imp userid=scott/tiger@ORCLYHY tables=(emp) file=d:\xxx.dmp ignore=y

【导入方案】

导入方案是指使用import工具将文件中的对象和数据导入到一个或是多个方案中。如果要导入其他方案，要求该用户具有dba的权限，或是imp\_full\_database

1. 导入自身的方案

imp userid=scott/tiger file=d:\xxx.dmp

1. 导入其他方案

要求该用户具有dba的权限

imp userid=system/tiger@ORCLYHY file=:\xxx.dmp fromuser=system touser=scott

【导入数据库】

在默认情况下，当导入数据库时，会导入所有对象结构和数据，案例如下：

imp userid=system/tiger@ORCLYHY full=y file=d:\xxx.dmp

【数据字典和动态性能视图】

【介绍】

数据字典是Oracle数据库中最重要的组成部分，它提供了数据库的一些系统信息。

动态性能视图记载了历程启动后的相关信息。

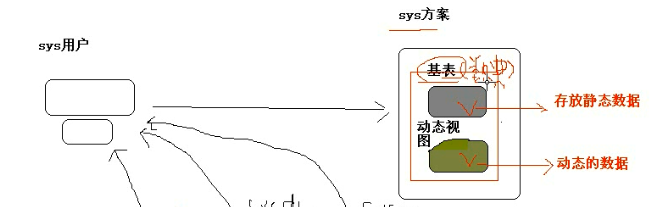
【数据字典】

数据字典记录了数据库的系统信息，它只读表和视图的集合，数据字典的所有者为sys用户。

用户只能在数据字典上执行查询操作（select语句），而其维护和修改是由系统自动完成的。

这里我们谈谈数据字典的组成：数据字典包括数据字典基表和数据字典视图，其中基表存储数据库的基本信息，普通用户不能直接访问数据字典的基表。数据字典视图是基于数据字典基表所建立的视图，普通用户可以通过查询数据字典视图取得系统信息。数据字典视图主要包括：

user\_xxx , all\_xxx , dba\_xxx三种类型。



1. user\_tables

用于显示当前用户所拥有的所有表，他只返回用户所对应方案的所有表

比如：select table\_name from user\_tables;

1. all\_tables

用于显示当前用户可以访问的所有表.它不仅会返回当前用户方案的所有表,还会返回当前用户可以访问的其他方案的表

比如：select table\_name from all\_tables;

1. dba\_tables

他会显示所有方案拥有的数据库表。但是查询这种数据库字典视图，要求用户必须是dba角色或是有select any table系统权限。

例如：当用户system用户查询数据字典视图dba\_tables时,会返回system,sys,scott...方案所对应的数据库表。

【用户名，权限，角色】

在建立用户时，Oracle会把用户的信息存放到数据字典中，当给用户授予权限或是角色时，Oracle会将权限和角色的信息存放到数据字典中。

通过查询dba\_users可以显示所有数据库用户的详细消息;

通过查询数据字典视图dba\_sys\_privs,可以显示用户所具有的系统权限;

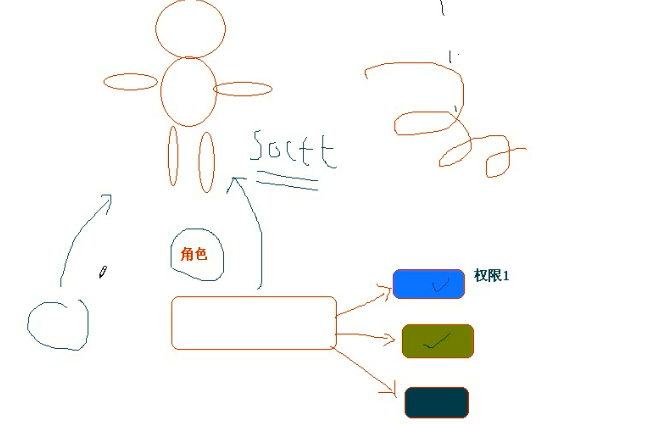
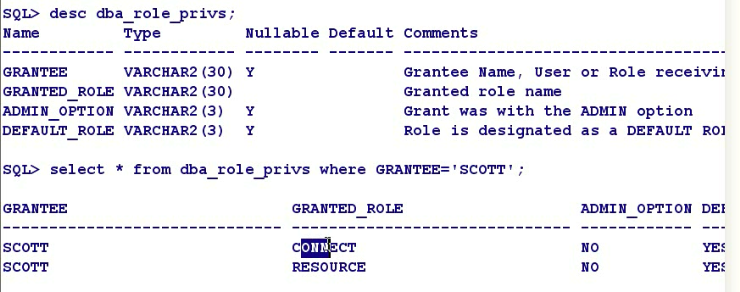
通过查询数据库字典视图dba\_tab\_privs可以显示用户具有的对象权限;

通过查询数据字典dba\_col\_privs可以显示用户具有的列权限;

通过查询数据字典dba\_col\_privs可以显示用户具有的列权限;

这里给大家在讲讲 角色和权限的关系.

例如:要查看scott具有的角色,可查询dba\_role\_privs.



？问题

1. 如何查询一个角色包括的权限？
   1. 一个角色包含的系统权限
   2. 一个角色包含的对象权限
2. Oracle究竟有多少种角色？
3. 如何查看某个用户具有什么样的角色

//查询Oracle中所有的系统权限，一般是dba

select \* from system\_privilege\_map order by name;

//查询Oracle中所有的角色,一般是dba

select \* from dba\_roles;

//查询Oracle中所有的对象权限,一般是dba

select distinct privilege from dba\_tab\_privs;

//查询数据库的表空间

select tablespace\_name from dba\_tablespaces;

//查询某个用户具有怎样的角色

select \* from dba\_role\_privs where grantee=’用户名’;

//查看某个角色包括哪些系统权限

select \* from role\_sys\_privs where grantee=’DBA’;或者是:select \* from role\_sys\_privs where role=’DBA’;

//查看某个角色包括的对象权限

select \* from dba\_tab\_privs where grantee=’角色名’;

【显示当前用户可以访问的所有数据字典视图】

select \* from dict where comments like ‘%grant%’;

【显示当前数据库的全称】

select \* from global\_name;

【其他说明】

数据字典记录有Oracle数据库的所有系统信息。通过查看数据字典可以取得以下系统信息：比如

1. 对象定义情况
2. 对象占用空间大小
3. 列信息
4. 约束信息

......

但是因为这些个信息，可以通过pl/sql developer工具查询到，所以这里我就飘过。

【动态性能视图】

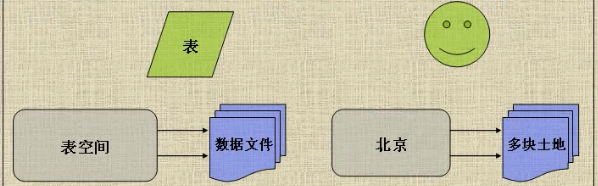
动态性能视图用于记录当前例程的活动信息，当启动Oracle server时，系统会建立动态性能视图；当停止Oracle server时，系统会删除动态性能视图。Oracle的所有动态性能视图都是以v\_$开始的，并且Oracle为每个动态性能视图都提供了相应的同义词，并且其同义词是以V$开始的，例如v\_$datafile的同义词为v$datafile；动态性能视图的所有者为sys，一般情况下，由dba或是特权用户来查询动态性能视图。

因为这个在实际中用的较少，所以飞过。

【管理表空间和数据文件】

【介绍】

表空间是数据库的逻辑组成部分。从物理上讲，数据库数据存放在数据文件中；从逻辑上讲，数据库则是存放在表空间中，表空间由一个或是多个数据文件组成。



【数据库的逻辑结构】

【介绍】

Oracle中逻辑结构包括表空间、段、区和快。

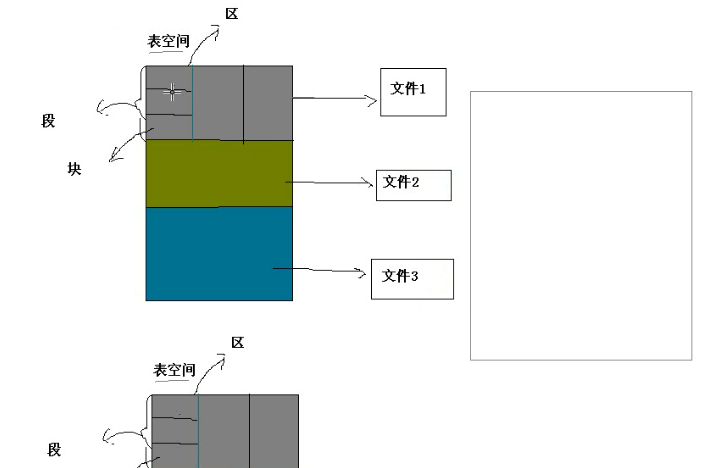
说明一下数据库由表空间构成，而表空间又是由段构成，二段又是由区构成，而区又是由Oracle块构成的这样的一种结构，可以提高数据库的效率。

为了让大家明白，我们画图说明逻辑关系：

【表空间】

表空间用于从逻辑上组织数据库的数据。数据库逻辑上是由一个或多个表空间组成的。通过表空间可以达到以下作用：

1. 控制数据库占用的磁盘空间
2. dba可以将不同数据类型部署到不同的位置，这样有利于提高i/o性能，同时利于备份和恢复等管理操作。



【建立表空间】

建立表空间是使用create tablespace命令完成的，需要注意是，一般情况下，建立表空间是特权用户或是dba来执行的，如果用其他用户来创建表空间，则用户必须要具有create tablespace的系统权限。

【建立数据表空间】

在建立数据库后，为了便于管理表，最好建立自己的表空间。

create tablespace data01 datafile ‘d:\data01.dbf’ size 20m uniform size 128k

说明:执行完上述命令后,会建立名称为data01的表空间,并为该表空间建立名称为data01.dbf的数据库文件,区的大小为128k。

【使用数据表空间】

create table mypart(deptno number(4),dname varchar2(14),loc varchar2(13)) tablespace data01;

【改变表空间的状态】

当建立表空间时，表空间处于联机（online）状态，此时该表空间是可以访问的，并且该表空间是可以读写的，即可查询该表空间的数据，而且还可以在表空间执行各种语句。但是在进行系统维护或是数据维护时，可能需要改变表空间的状态。一般情况下，由特权用户或是dba来操作。

1. 使用表空间脱机

alter tablespace 表空间名 offline；

1. 使用表空间联机

alter tablespace 表空间名 online；

1. 只读表空间

当建立表空间时，表空间可以读写，如果不希望在该表空间上执行update，delete，insert操作，那么可以将表空间修改为只读

alter tablespace 表空间名 read only；

【改变表空间的状态】

我们给大家举一个实例，说明只读特性：

1. 知道表空间名，显示该表空间包括的所有表

select \* from all\_tables where tablespace\_name=’表空间名’;

1. 知道表名，查看该表属于哪个表空间

select tablespace\_name,table\_name from user\_tables where table\_name=''表名’;--表名要大写

通过2我们可以知道scott.emp是在system这个表空间上,现在我们可以将system改为只读的但是我们不会成功,因为system是系统表空间,如果是普通表空间,那么我们就可以将其设为只读的.

给大家做一个演示,可以加强理解.

1. 使用表空间可读写

alter tablespace 表空间名 read writer;

1. 删除表空间

一般情况下，由特权用户或是dba来操作，如果是其他用户操作，那么要求用户具有drop tablespace系统权限。

drop tablespace ‘表空间名’ including contents and datafiles;

说明:including contents 表示删除表空间时,删除该空间的所有数据库对象,而datafiles表示将数据库文件也删除.

1. 扩展表空间

表空间是由数据文件组成的，表空间的大小实际上就是数据文件相加后的大小。那么我们可以想象，假定表employee存放到data01表空间上，初始大小就是2m,当数据满2m空间后,如果在向employee表插入数据,这样就会显示空间不足的错误.

案例说明:

1. 建立一个表空间 hy01
2. 在该表空间上建立一个普通表mydment其结构和dept一样
3. 向表中加入数据insert into mydment select \* from dept;
4. 当一定时候加入数据就会出现无法扩展的问题,怎么办?
5. 就扩展该表空间,为其增加更多的存储空间.有三种方法:

[扩展表空间]

1. 增加数据文件

SQL> alter tablespace data01 add datafile 'd:\data01.dbf' size 20m;

1. 增加数据文件的大小

SQL> alter tablespace datafile 'd:\data01.dbf' resize 20m;

这里需要注意的是数据文件的大小不要超过500m

1. 设置文件的自动增长

SQL> alter tablespace datafile 'd:\data01.dbf' autoextend on next 10m maxsize 500m;

6、移动数据文件(故障处理)

有时，如果你的数据文件所在的磁盘损坏时，该数据文件将不能再使用，为了能够重新使用，需要将这些文件的副本移动到其他磁盘，然后恢复。

下面一移动数据文件hy01.dbf为例来说明:

1. 确定数据文件所在的表空间

select table\_name from dba\_data\_files where file\_name=’d:\hy01.dbf’;

1. 使表空间脱机

确保数据文件的一致性,将表空间转变为offline的状态.

alter tablespace hy01 offline;

1. 使用命令移动数据文件到指定的目标位置

sql>host move d:\hy01.dbf c:\hy01.dbf;

1. 执行alter tablespace 命令

在物理上移动了数据后,还必须执行alter tablespace命令对数据库文件进行逻辑修改:

sql>alter tablespace hy01 rename datafile ‘d:\hy01.dbf’ to ‘c:\hy01.db’;

1. 使表空间联机

在移动了数据文件后,为了使用户可以访问该表空间,必须将其转变为online状态:

sql>alter tablespace data01 online;

1. 显示表空间信息

查询数据字典视图dba\_tablespaces，显示表空间的信息：

sql>select tablespace\_name fromdba\_tablespaces；

1. 显示表空间所包含的数据文件

查询数据字典视图dba\_data\_files，可显示表空间所包含的数据文件，如下：

sql>select file\_name, bytes from dba\_data\_files where tablespace\_name=’表空间名’;

【表空间小结】

1. 了解表空间和数据文件的作用
2. 掌握常用表空间，undo表空间和临时表空间的建立方法
3. 了解表空间的各个状态（online，offline，read write，read only）的作用，及如何改变表空间的状态的方法。
4. 了解移动数据文件的原因，及使用alter tablespace和alter datatable 命令移动数据文件的方法。

【其他表空间】

除了最常用的数据表空间外，还有其他类型表空间：

1. 索引表空间
2. undo表空间
3. 临时表空间
4. 非标准块的表空间

这几种表空间，大家可以自己参考书籍研究，这里我就不讲了。

【其他说明】

关于表空间的组成部分 段/区/块，我们在后边讲解

***第五部分 数据库维护***

【内容介绍】

1. 维护数据的完整性√
2. 管理索引√
3. 管理权限和角色√

【期望目标】

1. 掌握维护Oracle数据完整性的技巧
2. 理解索引概念，会建立索引
3. 管理Oracle的权限和角色

【维护数据的完整性】

【介绍】

数据的完整性用于确保数据库数据遵从一定的商业和逻辑规则。在Oracle中，数据完整性可以使用约束、触发器、应用程序（过程、函数）三种方法来实现，在这三种方法中，因为约束易于维护，并且具有最好的性能，所以作为维护数据完整性的首选。

【约束】

约束用于确保数据库数据满足特定的商业规则。

在Oracle中，约束包括：not null，unique，primary key，foreign key和check五种。

* not null（非空）：如果在列上定义了not null，那么当插入数据时，必须为列提供数据。
* unique（唯一）：当定义了唯一约束后，该列值是不能重复的。但是可以为null。
* primary key（主键）：用于唯一的标示行的数据，当定义主键约束后，该列不但不能重复而且不能为null。

需要说明的是：一张表最多只能有一个主键，但是可以有多个unique约束。

* foreign key（外键）：用于定义主表和从表之间的关系。外键约束要定义在从表上，主表则必须具有主键约束或是unique约束。当定义外键约束后，要求外键列数据必须在主表的主键列存在或是为null。
* check：用于强制行数据必须满足的条件，假定在sal列上定义了check约束，并要求sal列值在1000~2000之间，如果不在1000~2000之间就会提示出错。

【使用】

商店售货系统表设计案例：

现在有一个商店的数据库，记录客户及其购物情况，由下面三个表组成：

商品goods（商品号goodsID，商品名goodsname，单价unitprice，商品类别category，供应商provider）；

客户customer（客户号customerID，姓名name，住址address，电邮email，性别sex，身份证cardID）；

购买purchase（客户号customerID，商品号goodsID，购买数量nums）；

请用sql语句完成下列功能：

1. 建表，在定义中要求声明：
   1. 每个表的主外键；
   2. 客户的姓名不能为空值；
   3. 单价必须大于0，购买数量必须在1到30之间；
   4. 电邮不能够重复；
   5. 客户的性别必须是 男 或者 女 ，默认是男。
2. 如果在建表时忘记建立必要的约束，则可以在建表后使用altertable命令为表增加约束。但是要注意：增加not null约束时，需要使用modify选项，而增加其他四种约束使用add选项。
   1. 增加商品名也不能为空；
   2. 增加身份证也不重复；
   3. 增加客户的住址只能是‘海淀’、‘朝阳’、‘东城’、‘西城’、‘通州’、‘崇文’。

|  |
| --- |
| create table goods  (  goodsid char(8) primary key,--主键  goodsname varchar(30),  unitprice number(10,2) check(unitprice>0),  category varchar2(8),  provider varchar2(30)  );  create table customer  (  customerid char(8) primary key,--主键  name varchar2(50) not null,--不为空  address varchar2(50),  email varchar2(50) unique,  sex char(2) default '男' check(sex in('男','女')),  cardid char(18)  );  create table purchase  (  customerid char(8) references customer(customerid),  goodsid char(8) references goods(goodsid),  nums number(5) check(nums between 1 and 30)  );  alter table goods modify goodsname not null;  alter table customer add constraint uq\_customer\_cardid unique(cardid);  alter table customer add constraint addresscheck check(address in('东城','西城')); |

【删除约束】

当不再需要某个约束时，可以删除。

alter table 表名 drop constraint 约束名称；

\*\*特别说明：

再删除主键约束时，可能有错误，比如：

altertable 表名 drop primary key;

这是因为如果在两张表存在主从关系,那么在删除表的主键约束时,必须带上 cascade选项. 比如:

alter table 表名 drop primary key cascade;

【显示约束信息】

1. 显示约束信息

通过查询数据字典视图user\_constraints,可以显示当前用户所有的约束的信息。

select constraint\_name,constraint\_type,status,validated from user\_constraints where table\_name=’表名’;

1. 显示约束列

通过查询数据字典视图user\_cons\_columns,可以显示约束所对应的表列信息。

select column\_name，position from user\_cons\_columns where constraint\_name=’约束名’;

1. 当然也有更容易的方法,直接用pl/sql developer查看即可。

【列级定义】

列级定义是在定义列的同时定义约束。

如在department表定义主键约束

create table department4

(dept\_id number(2) constraint pk\_department primary key,name varchar2(12),loc varchar2(12));

【表级定义】

表级定义是指在定义了所有列后，在定义约束。这里需要注意：not null约束只能在列级上定义。

以在建立employee2表时定义主键约束和外键约束为例：

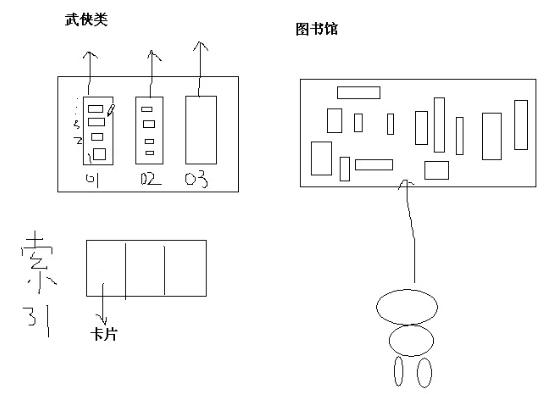
create table employee(emp\_id number(4),name varchar2(15),dept\_id number(2),constraint pd\_employee primary key(emp\_id),constraint fk\_department foreign key(dept\_id) references department4(dept\_id));

【管理索引】

【原理介绍】

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低i/o次数，从而提高数据访问性能。索引有很多种我们主要介绍常用的几种：

为什么添加了索引后，会加快查询速度呢？（和图书馆管理相似）



建索引不能随便的建，因为索引是需要专人的维护，如果不常用的字段不适合建立索引。

【创建索引】

【单列索引】

单列索引是基于单个列所建立的索引，比如：

create index 索引名 on 表名（列名）；

【复合索引】

复合索引是基于两列或是多列索引。在同一张表上可以有多个索引，但是要求列的组合必须不同，比如：

SQL>create index emp\_idx1 on emp（ename，job）；

SQL>create index emp\_idx1 on emp（job，ename）；

【使用原则】

①在大表上建立索引才有意义

②在where子句或是连接条件上经常引用的列上建立索引

③索引的层次不要超过4层

这里能不能给学生演示这个效果呢？

如何构建一个大表呢？

【索引的缺点】

索引有一些先天不足：

1. 建立索引，系统要占用大约为表的1.2倍的硬盘和内存空间来保存索引。
2. 更新数据的时候，系统必须要有额外的时间来同时对索引进行更新，以维持数据和索引的一致性。

实践表明，不恰当的索引不但于事无补，反而会降低系统性能。因为大量的索引在进行插入、修改和删除操作时比没有索引话费更多的系统时间。

比如在如下字段建立索引应该是不恰当的：

1. 很少或从不引用的字段；
2. 逻辑型的字段，如男或女，是或否等。宗山所述，提高查询效率是以消耗一定的系统资源为代价的，索引不能盲目的建立，这是考验一个DBA是否优秀的很重要的指标。

【其他索引】

【介绍】

按照数据存储方式,可以分为B\*树、反向索引、位图索引；

按照索引列的个数分类，可以分为单列索引、复合索引；

按照索引列值得唯一性，可以分为唯一索引和非唯一索引；

此外还有函数索引，全局索引，分区索引……

特别说明：

在不同的情况下我们会在不同的列上建立索引，甚至建立不同种类的索引，请记住，技术是死的，人是活的。比如：

B\*树索引建立在重复值很少的列上，而位图索引则建立在重复值很多、不同值相对固定的列上。

【显示索引信息】

【显示表的所有索引】

在同一张表上可以有多个索引，通过查询数据字典视图dba\_indexs 和 user\_indexs，可以显示索引信息。其中dba\_indexs用于显示数据库所有的索引信息，而user\_indexs用于显示当前用户的索引信息：

select index\_name，index\_type from user\_indexs where table\_name=‘表名’；

【显示索引列】

通过查询数据字段视图user\_ind\_columns，可以显示索引对应的列信息。

select table\_name，column\_name from user\_ind\_columns where index\_name=‘IND\_ENAME’;

你也可以通过ql/sql developer工具查看索引信息。

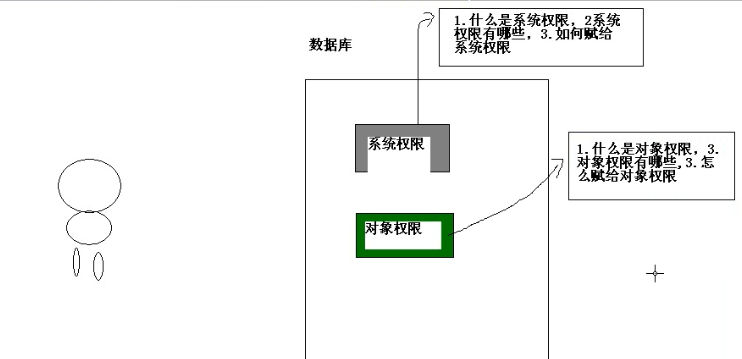
【管理权限和角色】

【介绍】

这一部分我们主要看看Oracle中如何管理权限和角色，权限和角色的区别在哪里。

当刚刚建立用户时，用户没有任何权限，也不能执行任何操作。如果要执行某种特定的数据库操作，则必须为其授予系统的权限；如果用户要访问其他方案的对象，则必须为其赋于对象的权限。为了简化权限的管理，可以使用角色。这里我们详细的介绍

角色的出现就是为了简化数据库权限的管理。



【权限】

权限是指执行特定类型sql命令或是访问其他方案对象的权利，包括系统权限和对象权限两种：

【系统权限】

【介绍】

系统权限是指执行特定类型sql命令的权利，它用于控制用户可以执行的一个或是一组数据库操作，比如当用户具有create table权限时，可以在其方案中建表，当用户具有create any table权限时，可以在任何方案中建表。Oracle提供了100多种系统权限。

常用的有：

create session 连接数据库

create table 建表

create view 建视图

create public synonym 建同义词

create procedure 建过程、函数、包

create trigger 建触发器

create cluster 建簇

【显示系统权限】

Oracle提供了100多种系统权限，而且Oracle的版本越高，提供的系统权限就越多，我们可以查询数据字典视图system\_privilege\_map，可以显示所有系统权限。

select \* from system\_privilege\_map order by name;

【授予系统权限】

一般情况下，授予系统权限是由dba完成的，如果用其他用户来授予系统权限，则要求该用户必须具有grant any privilege的系统权限在授予系统权限时，可以带有with admin option选项，这样被授予权限的用户或是角色还可以将该系统权限授予其他的用户或是角色。

为了快速理解，举例说明：

1. 创建两个用户 Ken，Tom。初始阶段他们没有任何权限，如果登录就会给出错误的信息

create user Ken identified by Ken；

1. 给用户Ken授权

①grant create session，create table to Ken with admin option；

②grant create view to Ken；

1. 给用户Tom授权

我们可以通过Ken给Tom授权，因为with admin option是加上的。当然也可以通过dba给Tom授权，我们就用Ken给Tom授权：

①grant create session ，create table to Tom；

②grant create view to Tom；→【不成功】

【回收系统权限】

一般情况下，回收系统权限时dba来完成的，如果其他用户来回收系统权限，要求该用户必须具有相应系统权限及转授系统权限的选项（with admin potion）。回收系统权限使用revoke来完成。

当回收了系统权限后，用户就不能执行相应的操作了，但是请注意，系统权限系统权限级联回收问题？【不是级联回收】

system-----------------→Ken-------------------→Tom

（create session）（create session）（create session）

用system执行如下操作：

revoke create session from Ken；请思考：Tom还能登陆吗？

【对象权限】

指访问其他方案对象的权利，用户可以直接访问自己方案的对象，但是如果要访问别的方案的对象，则必须具有对象的权限。

比如Smith用户要访问scott .emp表（scott方案emp表）。

则必须在scott .emp表上具有对象的权限。

常用的有：

alter 修改

delete 删除

select 查询

insert 添加

update 修改

index 索引

references 引用

execute 执行

【显示对象权限】

通过数据字段视图可以显示用户或是角色所具有的对象权限。视图为dba\_tab\_privs

sql>conn system/manager;

sql>select distinct privilege from dba\_tab\_privs;

sql>select grantor,owner ,table\_name,privilege from dba\_tab\_privs where grantee=’BLECK’;

【授予对象权限】

在Oracle9i前,授予对象权限是由对象的所有者来完成的,如果用其他的用户来操作,则需要用户具有相应的(with grant option)权限,从Oracle9i开始,dba用户(sys,system)可以将任何对象上的权限授予其他用户.授予对象权限是用grant命令来完成的。

对象权限可以授予用户，角色和public。在授予权限时，如果带有with grant option选项，则可以将该权限转授给其他用户。但是要注意 with grant option选项不能被授予角色。

1. monkey用户要操作scott.emp 的数据,则必须授予相应的对象权限

①希望monkey可以查询scott.emp的表数据,怎样操作?

SQL>grant select on emp to monkey;

②希望monkey可以修改scott.emp的表数据,怎样操作?

SQL>grant update on emp to monkey;

③希望monkey可以删除scott.emp的表数据,怎样操作?

SQL>grant delete on emp to monkey;

④有没有更加简单的方法,一次把所有权限赋给monkey?

SQL>grant all on emp to monkey;

1. 能否对monkey访问权限更加精细控制.(授予列权限)

①希望monkey只可以修改scott.emp的表的sal字段,怎样操作?

SQL>grant update on emp(sal) to monkey;

②希望monkey只可查询scott.emp的表的ename,sal数据,怎样操作?

SQL>grant select on emp(ename,sal) to monkey;

1. 授予alter权限

如果black用户要修改scott.emp表的结构,则必须授予alter对象权限

sql>conn scott/tiger;

sql>grant alter on emp to black;

当然也可以用system,sys来完成这件事.

1. 授予execute权限

如果用户想要执行其他方案的包/过程/函数,则需有execute权限.

比如为了让Ken可以执行包dbms\_transaction,可以授予execute权限

sql>conn system/tiger;

sql>grant execute on dbms\_transaction to ken;

1. 授予index权限

如果想在别的方案的表上建立索引,则必须具有index对象权限,如为了让black可以在scott.emp上建立索引,就给其index的对象权限.

sql>conn scott/tiger;

sql>grant index on scott.emp to black;

1. 使用with grant option选项

该选项用于转授对象权限.但是该选项只能被授予用户,而不能授予角色.

sql>conn scott/tiger;

sql>grant select on emp to black with grant option;

sql>conn black/tiger;

sql>grant select on scott.emp to jones;

【回收对象权限】

在Oracle9i，回收对象的权限可以有对象的所有者来完成，也可以用dba用户（sys，system）来完成。这里要说明的是：收回对象权限后：用户就必能执行相应的sql命令，但是要注意的是对象的权限是否会被级联收回？【级联收回】

如：

scott------------------------→black-----------------------→Jones

（select on emp） （select on emp） （select on emp）

sql>conn scott/tiger@accp;

sql>revoke select on emp from black;

请思考:Jones能否查询scott.emp表数据.

【角色】

【介绍】

角色就是相关权限的命令集合，使用角色的主要目的就是为了简化权限的管理。假定有用户a、b、c为了让他们都拥有权限。

①连接数据库。

②在scott.emp表上select,insert,update。

如果采用直接授权操作,则需要进行12次授权.[因为要进行12次授权操作,所以比较麻烦,怎么办?]

我们如果采用角色就可以简化:

首先将create session,select on scott.emp,insert on scott.emp, update on scott.emp授予角色,然后将该角色授予a、b、c用户，这样就可以三次授权搞定。角色分为预定义和自定义角色两类：【可以考虑使用自定义角色来解决问题的】

【预定义角色】

预定义角色是指Oracle所提供的角色，每种角色都用于执行一些特定的管理任务，下面我们介绍常用的预定义角色connect，resource，dba。

1. connect

connect角色具有一般应用开发人员需要的大部分权限，当建立了一个用户后，多数情况下，只要给用户授予connect和resource角色就够了，那么connect角色具有哪些系统权限呢？

alter session，create cluster，create database link，create session，create table，create view，create sequence

1. resource

resource角色具有应用开发人员所需要的其他权限，比如建立存储过程、触发器等。这里需要注意的是resource角色隐含了unlimited tablespace系统权限。

resource角色包含以下系统权限：

create cluster

create indextype

create table

create sequence

create type

create procedure

create trigger

1. dba

dba角色具有所有的系统权限，及with admin option选项，默认的dba用户为sys和system他们可以将任何系统权限授予其他用户。但是要注意的是dba角色不具备sysdba和sysoper的特权（启动和关闭数据库）

【自定义角色】

顾名思义就是自己定义的角色，根据自己的需要来定义。一般是dba来建立，如果用别的用户来建立，则需要具有create role的系统权限。在建立角色时可以指定验证方式（不验证，数据库验证等）

1. 建立角色（不验证）

如果角色是公用的角色，可以采用不验证的方式建立角色。

create role 角色 not identified；

1. 建立角色（数据库验证）

采用这样的方式时，角色名、口令存放在数据库中。当激活该角色时，必须提供口令。在建立这种角色时，需要为其提供口令

create role 角色名 identified by bruce；

1. 角色授权

当建立角色时，角色没有任何权限，为了使得角色完成特定任务，必须为其授予相应的系统权限和对象权限。

1. 给角色授权

给对象授予权限和用户授权没有太多区别，但是要注意，系统权限的unlimited tablespace和对象权限的with grant option选项是不能授予角色的。

sql>conn system/tiger;

sql>grant create session to 角色名 with admin option;

sql>conn scott/tiger@ORCLYHY;

sql>grant select on scott.emp to 角色名;

sql>grant insert,update,delete on scott.emp to 角色名

通过上面的步骤,就给角色授权了.

2.分配角色给某个用户

一般分配角色是由dba来完成的，如果要以其他用户身份分配角色，则要求用户必须具有grant any role的系统权限。

sql>conn system/tiger;

sql>grant 角色名 to black with admin option;

因为我给了with admin option选项

所以black可以把system分配给它的角色分配给别的用户.

1. 删除角色

使用drop role，一般是dba来执行，如用其他用户则要求该用户具有drop any role系统权限

sql>conn system/tiger;

sql>drop role 角色名;

如果角色被删除,那么black是否还可以登录?

1. 删除myrole1角色

drop role myrole1

1. conn black/tiger;

经验证,无法登录.

1. 显示角色信息

①显示所有角色

sql>select \* from dba\_roles;

②显示角色具有的系统权限

sql>select privilege,damin\_option from role\_sys\_privs where role=’角色名’;

③显示角色具有的对象权限

通过查询数据字典视图dba\_tab\_privs可以查看角色具有的对象权限或是列的权限.

④显示用户具有的角色,及默认角色

当用户的身份连接到数据库时,Oracle会自动的激活默认的角色,通过查询数据字典视图dba\_role\_privs可以显示某个用户具有的所有角色及当前默认的角色

sql>select granted\_role,default\_role from dba\_role\_privs where grantee=’用户名’;

1. 精细访问控制

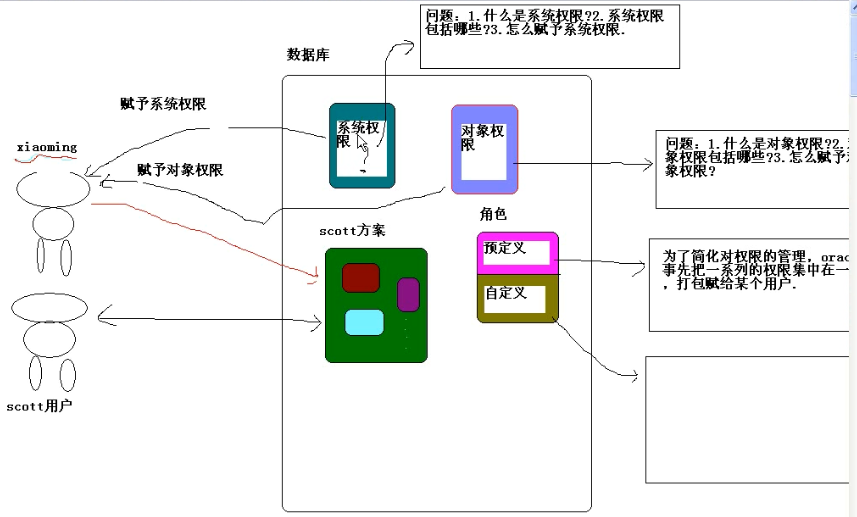
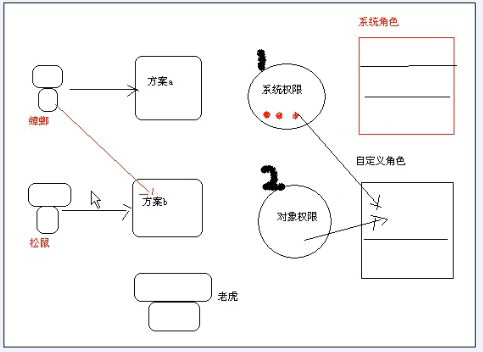
是指用户可以使用函数、策略实现更加细微的安全访问控制。如果使用精细访问控制，则当在客户端发出sql语句（select，insert，update，delete）时，Oracle会自动在sql语句后追加谓词（where子句），并执行新的sql语句。通过这样的控制，可以使得不同的数据库用户在访问相同表时，返回不同的数据信息，如下：

用户：scott black Jones

策略 emp\_access

数据库表emp

如上所示：通过策略emp\_access,用户scott,black,Jones在执行相同的sql语句时,可以返回不同的结果.例如,当执行select ename from emp;时,更具实际情况可以返回不同的结果.



***第六部分 PL/SQL编程***

【内容介绍】

1. pl/sql的介绍
2. pl/sql的基础

【期望目标】

1. 理解Oracle的pl/sql概念
2. 掌握pl/sql编程技术（包括编写存储过程、函数、触发器……）

【pl/sql是什么】

pl/sql（procedure language/sql）是Oracle在标准的SQL语言上的扩展。pl/sql不仅允许嵌入SQL语言，还可以定义变量和常量，允许使用条件语句和循环语句，允许使用例外处理各种错误，这样使得它的功能变得更加强大。

1.过程，函数，触发器是pl/sql编写的

2.过程,函数,触发器是在Oracle中的

3.pl/sql是非常强大的数据库过程语言

4.过程,函数可以在Java程序中调用

【学习的必要性】

1. 提供应用程序的运行性能
2. 模块化的设计思想【分页的过程，订单的过程，转账的过程……】
3. 减少网络传输量
4. 提高安全性

【不好的一面】

1. 移植性不好。

【用什么编写pl/sql】

【sqlplus开发工具】

sqlplus是Oracle公司提供的一个工具，这个因为我们在以前介绍过的：

举一个简单案例：

编写一个存储过程，该过程可以向某表中添加记录。

|  |
| --- |
| --1.创建一个简单的表  create table mytest(name varchar2(30),passwd varchar2(30));  --2.创建过程  --replace表示如果有hy\_pro1,就替换  create or replace procedure hy\_pro1 is  --开始语句  begin  --执行部分  insert into mytest('于海洋','tiger');  --结束语句  end;  --创建存储过程  /  --如何查看错误信息  show error;  --将sql语句修改  insert into mytest values('于海洋','tiger');  --如何调用该过程  /\*  ①exec 过程名(参数值1,参数值2...);  ②call 过程名(参数值1,参数值2...);  \*/ |

【pl/sql developer开发工具】

pl/sql developer是用于开发pl/sql块的集成开发环境，它是一个独立的产品，而不是Oracle的一个附带品。

举一个简单案例：

编写一个存储过程，该过程可以删除某表记录。

|  |
| --- |
| --案例  create or replace procedure hy\_pro2 is  begin  --执行部分  delete from mytest where name='于海洋';  end;  / |

【介绍】

开发人员使用pl/sql编写应用模块时，不仅需要掌握sql语句的编写方法，还要掌握pl/sql语句及语法规则。pl/sql编程可以使用变量和逻辑控制语句，从而可以编写非常有用的功能模块。

比如：

分页存储过程模块、订单处理存储过程模块、转账存储过程模块……而且如果使用pl/sql编程，我们可以轻松的完成非常复杂的查询要求。【等学习了pl/sql编程基础再讲吧】

【pl/sql可以做什么】

【简单分类】

块（编程）：过程（存储过程），函数，触发器，包；

【编写规范】

①注释

单行注释 --

select \* from emp where empno=7788;--取得员工信息

多行注释

/\*...\*/来划分

②标识符号的命名规则

1. 当定义变量时,建议用v\_作为前缀v\_sal
2. 当定义常量时,建议用c\_作为前缀c\_rate
3. 当定义游标时,建议用\_cursor作为后缀 emp\_cursor;
4. 当定义例外时,建议用e\_作为前缀e\_error。

【pl/sql基础】

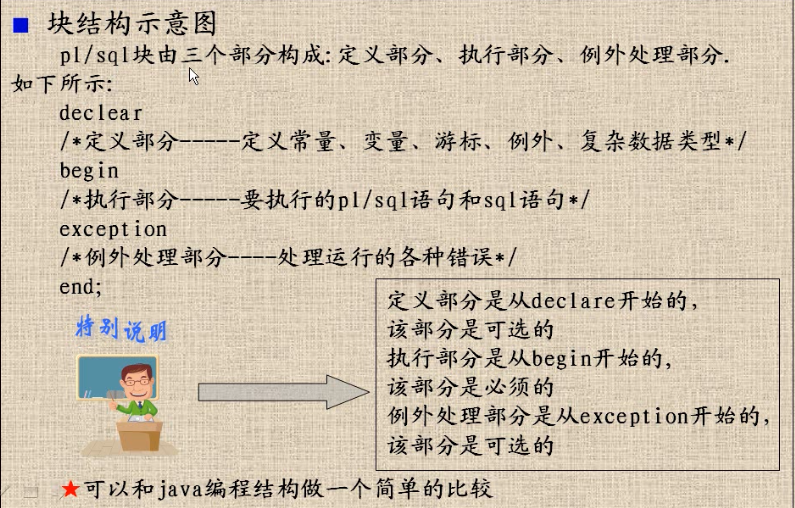
【pl/sql块】

【介绍】

块（block）是pl/sql的基本程序单元，编写pl/sql程序实际上就是编写pl/sql块。要完成相对简单的应用功能，可能只需要编写一个pl/sql块；但是如果要想实现复杂的功能，可能需要在一个pl/sql块中嵌套其他的pl/sql块

【块结构示意图】

pl/sql块由三个部分构成：定义部分、执行部分、例外处理部分。

如下所示：

declear

/\*定义部分-----定义常量\变量\游标\例外\复杂数据类型\*/

begin

/\*执行部分------要执行的pl/sql语句和sql语句\*/

exception

/\*例外处理部分------处理运行的各种错误\*/

end;

\*\*可以和Java编程结构做一个简单的比较.

Java程序的结构

public static voidmain(String[] args){

int a=1;

try{

a++

}catch(Exception e){

//异常捕获

}

}

【pl/sql块的实例】

实例1-只包括执行部分的pl/sql块

set serveroutput on --打开输出选项

begin

dbms\_output.put\_line(‘hello’);

end;

§相关说明:

dbms\_output是Oracle所提供的包(类似Java的开发包),包 包含一些过程,put\_line就是dbms\_output包的一个过程.

实例2-包含定义部分和执行部分的pl/sql块

|  |
| --- |
| *--有定义和执行部分的块*  *--把雇员的工资也显示出来*  declare  v\_ename varchar2(5);*--定义字符串变量*  v\_sal number(7,2);  begin  *--执行部分*  select ename,sal into v\_ename,v\_sal from emp where empno=&no;  *--在控制台显示雇员名*  dbms\_output.put\_line('雇员名:'||v\_ename ||'工资:'||v\_sal);  end;  / |

§相关说明：

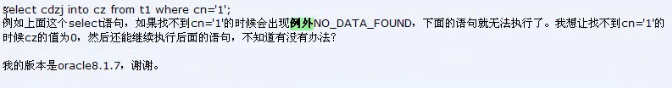
& 表示要接收从控制台输入的变量。

案例3-包括定义部分、执行部分和例外处理部分

为了避免pl/sql程序的运行错误，提高pl/sql的健壮性，应该对可能的错误进行处理，这个很有必要：

①比如在实例2中，如果输入了不存在的雇员号，应当做例外处理。

②有时出现异常，希望用另外的逻辑处理，



我们看看如何完成①的要求

|  |
| --- |
| *--有定义和执行部分的块*  *--把雇员的工资也显示出来*  declare  v\_ename varchar2(5);*--定义字符串变量*  v\_sal number(7,2);  begin  *--执行部分*  select ename,sal into v\_ename,v\_sal from emp where empno=&no;  *--在控制台显示雇员名*  dbms\_output.put\_line('雇员名:'||v\_ename ||'工资:'||v\_sal);  *--异常处理*  exception  when no\_data\_found then  dbms\_output.put\_line('朋友,你的编号输入有误,请重新输入!');  end;  / |

相关说明：Oracle事先预定义了一些例外，no\_data\_found就是找不到数据的例外。

【过程】

过程用于执行特定的操作。当建立过程时，既可以指定输入参数（in），也可以指定输出参数（out）。在过程中通过使用输入参数，可以将数据传递到执行部分；通过使用输出参数，可以将执行部分的数据传递到应用环境。在sqlplus中可以使用create procedure命令来建立过程。

实例如下：

①请考虑编写一个过程，可以输入雇员名，薪资，可修改雇员的工资。

|  |
| --- |
| *--案例4*  createprocedure hy\_pro3(hyName varchar2,newSal number)is  begin  *--执行部分,根据用户名去修改工资*  update emp set sal=newSal where ename=hyName;  end;  /  *--调用过程*  call hy\_pro3('SCOTT',4567); |

②如何调用过程有两种方法：

exec……

call ……

③如何在Java程序中调用一个存储过程

|  |
| --- |
| *--演示Java程序去调用Oracle的存储过程案例*  import java.sql.\*;  ...  main(String[] args){  try{  //1.加载驱动  forName("");  //2.  Connection ct=DriverManager.getConnection("");  //3.创建CallableStatement  CallableStatement cs=ct.prepareCall("{call hy\_pro3(?,?)}")  //4.给问号赋值  cs.setString(1,"SMITH");  cs.setString(2,10);  //5.执行  cs.execute();  //6.关闭资源  cs.close();  ct.close()  }catch(Exception e){  e.printStackTrace();  }  } |

？如何使用过程返回值。

特别说明：

对于过程我们会在以后给大家详细具体的介绍，现在请大家先有一个概念。

【函数】

函数用于返回特定的数据，当建立函数时，在函数头部必须包含return子句，而在函数体内必须包含return语句返回的数据。我们可以使用create function来建立函数。

实际案例：

|  |
| --- |
| *--函数案例*  *--输入雇员的姓名,返回该雇员的年薪*  createfunction hy\_fun1(hyName varchar2)returnnumberis  yearSal number(7,2);  begin  *--执行部分*  select sal \*12+nvl(comm,0)\*12  into yearSal  from emp  where ename = hyName;  return yearSal;  end;  / |

在sqlplus中调用函数

sql>var income number;

sql>call hy\_fun1(‘SCOTT’) into:income;

sql>print income;

同样我们可以在Java程序中调用该函数

select hy\_fun1(‘SCOTT’) from dual;//这样可以通过rs.getInt(1)得到返回结果

【包】

包用于在逻辑上组合过程和函数，它由包规范和包体两部分组成。

①我们可以使用create package 命令来创建包：

实例：

--创建一个包,包名为hy\_package

--声明该包有一个过程update\_sal

--声明该包有一个函数annual\_income

create package hy\_package is

procedure update\_sal(name varchar2, newSal number);--过程的声明

function annual\_income(name varchar2) return number;--函数的声明

end;

包的规范只包含了过程和函数的说明，但是没有过程和函数的实现代码。包体用于实现包规范中的过程和函数。

②建立包体可以使用create package body命令

--给包hy\_package实现包体

|  |
| --- |
| createpackagebody hy\_package is  procedure update\_sal(namevarchar2, newsal number)is  begin  update emp set sal = newsal where ename =name;  end;  function annual\_income(namevarchar2)returnnumberis  annual\_salary number;  begin  select sal \*12+nvl(comm,0)  into annual\_salary  from emp  where ename =name;  return annual\_salary;  end;  end; |

③如何调用包的过程或是函数

当调用包的过程或是函数时，在过程和函数前需要带有包名，如果要访问其他方案的包，还需要在包名前加方案名。

如：

SQL>call hy\_package.update\_sal(‘SCOTT’,1500);

特别说明:

包是pl/sql中非常重要的部分,我们在使用过程分页时,将会再次体验它的威力。

【触发器】

触发器是指隐含的执行的存储过程。当定义触发器时，必须要指定触发的事件和触发的操作，常用的触发事件包括insert，update，delete语句，而触发器实际就是一个pl/sql块。可以使用create trigger来建立触发器。

特别说明：

我们会在后面详细为大家介绍触发器的使用，因为触发器是非常有用的，可维护数据库的安全和一致性。

【定义并使用变量】

【介绍】

在编写pl/sql程序时，可以定义变量和常量；在pl/sql程序中包括有：

①标量类型（scalar）

②复合类型（composite）

③参照类型（reference）

④lob（large object）

【标量】---常用类型

在编写pl/sql块时，如果要使用变量，需要在定义部分定义变量。pl/sql中定义变量和常量的语法如下：

identifier [constant] datatype [not null] [:=| default expr]

identifier:名称

constant:指定常量.需要指定它的初始值,且其值是不能改变的.

datatype:数据类型

not null:指定变量值不能为null

:= 给变量或是常量指定初始值

default:用于指定初始值

expr:指定初始值的pl/sql表达式,可是文本值、其他变量、函数等

举例说明：

【标量定义的案例】

①定义一个变长字符串

v\_ename varchar2(20);

②定义一个小数 范围-9999.99~9999.99

v\_sal number(6,2);

③定义一个小数并给一个初始值为5.4 :=是pl/sql的赋值号

v\_sal number(6,2):=5.4;

④定义一个日期类型的数据

v\_hiredate date;

⑤定义一个布尔变量,不能为null,初始值为false

v\_valid boolean not null default false;

【标量】---使用标量

在定义号变量后，就可以使用这些变量。这里需要说明的是pl/sql块为变量赋值不同于其他的编程语言，需要在等号前加冒号（:=）

下面以输入员工号,显示雇员姓名、工资、个人所得税（税率为0.03）为例。说明变量的使用，看看如何编写？

案例说明：

|  |
| --- |
| *--案例*  declare  c\_tax\_rate number(3,2):=0.03;  *--用户名*  v\_ename varchar2(5);  v\_sal number(7,2);  v\_tax\_sal number(7,2);  begin  *--执行*  select ename,sal into v\_ename,v\_sal from emp where empno=&no;  *--计算所得税*  v\_tax\_sal:=v\_sal\*c\_tax\_rate;  *--输出*  dbms\_output.put\_line('姓名是:'||v\_ename||'工资:'||v\_sal||'交税:'||v\_tax\_sal);  end;/ |

【标量】--使用%type类型

对于上面的pl/sql块有一个问题：

就是如果员工的姓名超过了5个字符的话，就会有错误，为了降低pl/sql程序的维护工作量，可以用%type属性定义变量,这样他会按照数据库列来确定你定义的变量的类型和长度.

我们看看这个怎么使用:

标识符名 表名.列名%type;

|  |
| --- |
| *--案例*  declare  c\_tax\_rate number(3,2):=0.03;  *--用户名*  **v\_ename emp.ename%type;**  v\_sal number(7,2);--这个也可以改动  v\_tax\_sal number(7,2);  begin  *--执行*  select ename,sal into v\_ename,v\_sal from emp where empno=&no;  *--计算所得税*  v\_tax\_sal:=v\_sal\*c\_tax\_rate;  *--输出*  dbms\_output.put\_line('姓名是:'||v\_ename||'工资:'||v\_sal||'交税:'||v\_tax\_sal);  end;/ |

【复合变量】

【介绍】

用于存放多个值得变量。主要包括这几种：

①pl/sql记录

②pl/sql表

③嵌套表

④varray

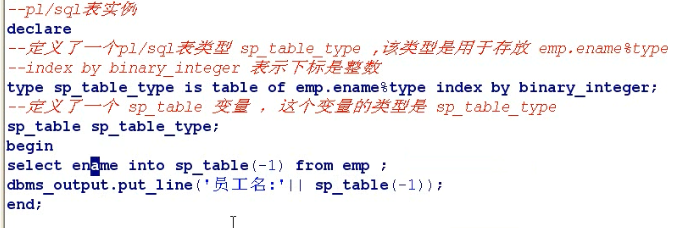
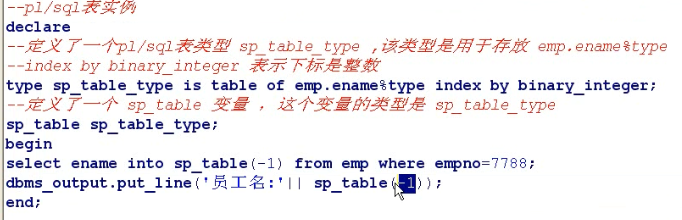
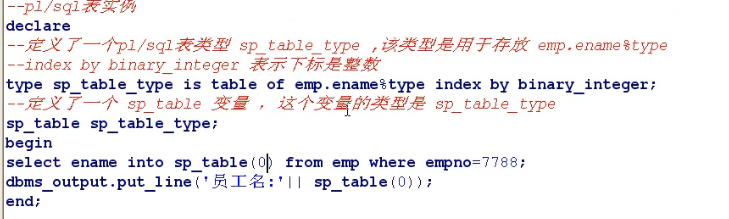
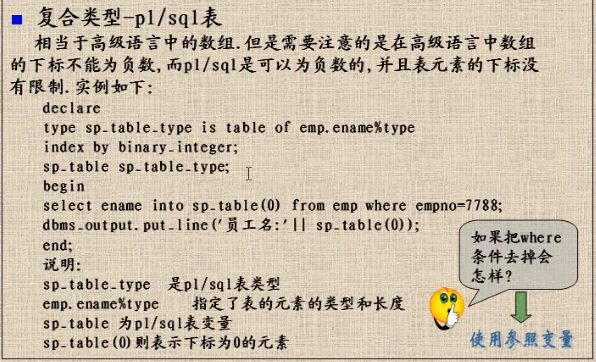
【复合类型】-- pl/sql记录

类似于高级语言中的结构体，需要注意的是，当引用pl/sql记录成员时，必须要加记录变量作为前缀（记录变量，记录成员）如下：

|  |
| --- |
| --pl/sql记录实例  declare  --定义一个pl/sql记录类型emp\_record\_type,类型包含三个数据name,salary,title  type emp\_record\_type is recoord(name emp.ename%type,salary emp.sal%type,title emp.job%type);  --定义了一个hy\_record变量,这个变量的类型是emp\_record\_type  hy\_record emp\_record\_type;  begin  select ename,sal,job into hy\_record from where empno=7788;  dbms\_output.put\_line('员工名:'||emp\_record.name);  end; |

【复合类型】--pl/sql表

相当于高级语言中的数组。但是需要注意的是在高级语言中数组的下标不能为负数，而pl/sql是可以为负数的，并且表元素的下标没有限制。实例如下：



【嵌套表（nested table）】

【变长数组（varray）】--自己看书学习

【参照变量】

【介绍】

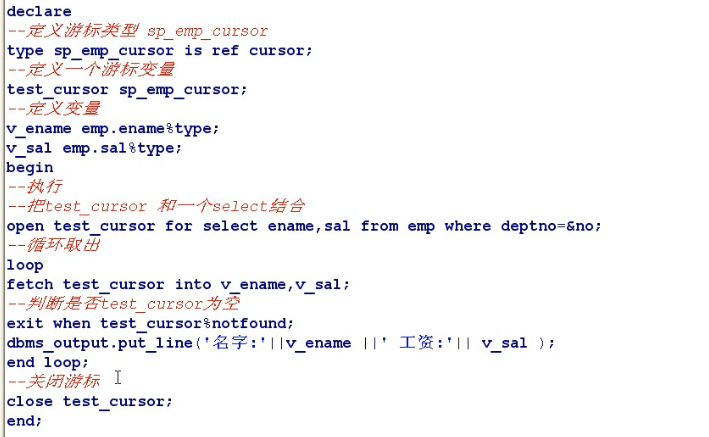
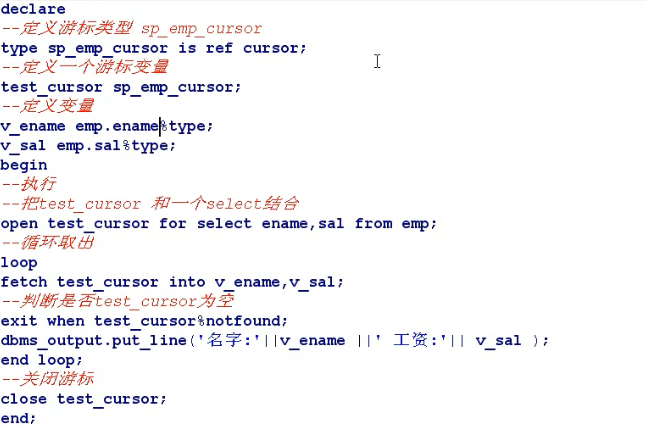
参照变量是指用于存放数值指针的变量。通过使用参照变量，可以使得应用程序共享相同对象，从而降低占用的空间。在编写pl/sql程序时，可以使用游标变量（ref cursor）和对象类型变量(ref obj\_type)两种参照变量类型

【参照变量】--ref cursor游标变量

使用游标时，当定义游标时不需要指定响应的select语句，但是当使用游标时（open时）需要指定select语句，这样一个游标就与一个select语句结合了。实例如下：

①请使用pl/sql编写一个块，可以输入部门号，并且显示该部门所有员工姓名和他的工资。

②在①的基础上，如果某个员工的工资低于200元，就增加100元。



***第七天***

【内容介绍】

1. pl/sql的进阶
2. Oracle的视图
3. Oracle的触发器

【期望目标】

1. 掌握pl/sql的高级用法（能编写分页过程模块，下订单过程模块……）
2. 会处理Oracle常见的例外
3. 会编写Oracle各种触发器
4. 理解视图的概念并能灵活使用视图

【控制结构】

【介绍】

在任何计算机语言(c,java,pascal)都有各种控制语句(条件语句,循环语句,顺序控制结构...)在pl/sql中也存在这样的控制结构。

在本部分学习完毕后，希望大家达到：

1. 使用各种if语句
2. 使用循环语句
3. 使用控制语句----goto和null

【条件分支语句】

pl/sql中提供了三种条件分支语句 if -- then，if -- then --else，if -- then -- elsif -- else

\*这里我们可以和Java语句进行一个比较

【简单的条件判断】 if -- then

？编写一个过程，可以输入一个雇员名，如果该雇员的工资低于2000，就给该雇员工资增加10%

|  |
| --- |
| create or replace procedure hy\_pro1(hyName varchar2) is  --定义  v\_sal emp.sal%type;  begin  --执行  select sal into v\_sal from emp where ename=hyName;  --判断  if v\_sal<2000 then  update emp set sal =sal\*1.1 where ename=hyName;  end if;  end;  /  --调用过程  exec hy\_pro1('SCOTT');  exec hy\_pro1('FORD'); |

【二重条件分支】 if -- then -- else

？编写一个过程，可以输入一个雇员名，如果雇员的补助不是0，就在原来的基础上增加100；如果补助为0就把补助设为200；

|  |
| --- |
| create or replace procedure hy\_pro1(hyName varchar2) is  --定义  v\_comm emp.comm%type;  begin  --执行  select comm into v\_comm from emp where ename=hyName;  --判断  if v\_comm<>0 then--<>表示不等于  update emp set comm =comm+100 where ename=hyName;  else  update emp set comm=nvl(comm,0)+200 where ename=hyName;  end if;  end;  /  --调用过程  exec hy\_pro1('SCOTT');  exec hy\_pro1('FORD'); |

【多重条件分支】 if -- then -- elsif -- else

？编写一个过程，可以输入一个雇员编号，如果该雇员的职位是PRESIDENT 就给他的工资增加1000，如果该雇员的职位是MANAGER就给他的工资增加500，其他职位的雇员工资增加200.

|  |
| --- |
| createorreplaceprocedure hy\_pro1(hyNo number)is  *--定义*  v\_job emp.job%type;  begin  *--执行*  select job into v\_job from emp where empno=hyNo;  *--判断*  if v\_job='PRESIDENT'then  update emp set sal =sal+1000where empno=hyNo;  elsif v\_job='MANAGER'  update emp set sal=sal+500where empno=hyNo;  else  update emp set sal=sal+200where empno=hyNo;  endif;  end;  /  *--调用过程*  exec hy\_pro1(7788);  exec hy\_pro1(7782);  exec hy\_pro1(7839); |

【控制结构】

【循环语句】--loop

是pl/sql中最简单的循环语句，这种循环语句以loop开头，以end loop结尾，这种循环至少会执行一次。相当于Java中的do while。

案例：现有一张表users，表结构如下

|  |  |
| --- | --- |
| 用户ID | 用户名 |
|  |  |

请编写一个过程，可输入用户名，并循环添加10个用户到users表中，用户编号从1开始增加。

|  |
| --- |
| createorreplaceprocedure hy\_pro1(spName varchar2)is  *--定义*  v\_num number:=1;  begin  loop  insertintousersvalues(v\_num,hyName);  *--判断是否要退出循环*  exitwhen v\_num=10;  *--自增*  v\_num:=v\_num+1;  endloop;  end;  / |

【循环语句】-- while循环

基本循环至少要执行循环体一次，而对于while循环来说，只有条件为true时,才会执行循环体语句,while循环以while...loop开始,以 end loop结束

案例:题干同上

请编写一个过程,可输入用户名,并循环添加10个用户到users表中,用户编号从11开始增加.

|  |
| --- |
| createorreplaceprocedure hy\_pro1(spName varchar2)is  *--定义*  v\_num number:=11;  begin  while v\_num<=20loop  insertintousersvalues(v\_num,hyName);  *--自增*  v\_num:=v\_num+1;  endloop;  end;  / |

【循环语句】 -- for循环

基本for循环的基本结构如下：

begin

for i in reverse 1..10 loop

insert into users values(i,'海洋');

end loop;

end;

/

我们可以看到控制变量i,在隐含中就在不停的增加.

【顺序控制语句】 -- goto,null

①goto语句

goto语句用于跳转到特定标号去执行语句.注意由于使用goto语句会增加程序的复杂性,并使得应用程序可读性变差,所以在做一般应用开发时,建议大家不要使用goto语句.

基本语法如下 goto lable,其中lable是已经定义好的标号名.

|  |
| --- |
| --goto案例  declare  i int :=1;  begin  loop  dbms\_output.put\_line('输出i='||i);  if i=10 then  goto end\_loop;  end if;  i:=i+1;  end loop;  <<end\_loop>>--这是标号  dbms\_output.put\_line('循环结束');  end;  / |

②null语句

null语句不会执行任何操作,并且会直接将控制传递到下一条语句.使用null语句的主要好处是可以提高pl/sql的可读性.

|  |
| --- |
| *--null案例*  declare  v\_sal emp.sal%type;  v\_ename emp.ename%type;  begin  select ename,sal into v\_ename,v\_sal from emp where empno=&no;  if v\_sal<3000then  update emp set comm=sal\*0.1where ename=v\_ename;  else  null;  endif;  end;  / |

【pl/sql进阶】

【编写分页过程】

【介绍】

分页是任何一个网站（BBS，网上商城。blog）都会使用到的技术，因此学习pl/sql编程开发就一定要掌握该技术。

【无返回值的存储过程】

古人云：欲速则不达，为了比较容易接受分页过程编写，我还是从简单到复杂，循序渐进的给大家讲解。首先是掌握最简单的存储过程，无返回值的存储过程：

案例：有一张表 book ，表结构如下：

|  |  |  |
| --- | --- | --- |
| 书号 | 书名 | 出版社 |
|  |  |  |

请编写一个过程，可以向book表添加书，要求通过Java程序调用该过程。

|  |
| --- |
| *--建表 book表*  createtable book(bookID number,bookName varchar(50),publishHouse varchar2(50));  *--编写过程*  *--in 表示输入参数,默认就是in*  *--out表示是输出参数*  createorreplaceprocedure hy\_pro1(hyBookId innumber,hyBookName invarchar2,hyPublishHouse invarchar2)is  begin  insertinto book values(hyBookId,hyBookName,hyPublishHouse)  end;  /  *--在Java中调用一个无返回值得过程*  publicclass A{  publicstatic void main(String[] args){  try{  //1.加载驱动  Class.forName("");  //2.连接数据库  Connection ct=DriverManager("","","");  //3.创建CallableStatement  CallableStatement cs=ct.prepareCall("{call hy\_pro1(?,?,?)}");  //4.给?赋值  cs.setString(1,"");  cs.setString(1,"");  cs.setString(1,"");  }catch(){}finally{//关闭各种资源}  }  } |

【有返回值的存储过程】-- 非列表

再看如何处理有返回值的存储过程：

案例：编写一个过程，可以输入雇员的编号，返回该雇员的姓名。

create or replace procedure hy\_pro1(hyNo in varchar2,hyName out varchar2) is

begin

select ename into hyName from emp where empno=spNo;

end;

/

--java 中执行

CallableStatement cs=ct.prepareCall("{call hy\_pro1(?,?)}");

--给?赋值

cs.setInt(1,7788);

cs.registerOutParameter(2,oracle.jdbc.OracleType.VARCHAR);

--执行

cs.execute();

--取出返回值,要注意?的顺序

String name=cs.getString(2);

案例扩展：编写一个过程，可以输入员工的编号，返回该雇员的姓名、工资、和岗位。

create or replace procedure hy\_pro1(hyNo in varchar2,hyName out varchar2,hySal out number,hyJob out varchar2) is

begin

select ename,sal,job into hyName,hySal,hyJob from emp where empno=hyNo;

end;

/

--java 中执行

CallableStatement cs=ct.prepareCall("{call hy\_pro1(?,?,?,?)}");

--给?赋值

cs.setInt(1,7788);

cs.registerOutParameter(2,oracle.jdbc.OracleType.VARCHAR);

cs.registerOutParameter(3,oracle.jdbc.OracleType.DOUBLE);

cs.registerOutParameter(4,oracle.jdbc.OracleType.VARCHAR);

--执行

cs.execute();

--取出返回值,要注意?的顺序

String name=cs.getString(2);

【有返回值的存储过程】-- 列表（结果集）

案例：编写一个过程，输入部门号，返回该部门所有雇员信息。对该题分析如下：

由于Oracle存储过程没有返回值，他的所有返回值都是通过out参数来替代的，列表同样也不例外，但由于是集合，所以不能用一般的参数，必须要用package了。所以要分两部分：

①建一个包，在该包中定义一个类型 test\_cursor，是个游标。如下：

create or replace package testpackage as

type test\_cursor is ref cursor；

end testpackage；

②建立存储过程。如下：

create or replace procedure testc(myno in number,p\_cursor out testpackage.test\_cursor) is

begin

open p\_cursor for select \* from emp where deptno=no;

end testc;

③下面看看如何在Java程序中调用.

CallableStatement cs=ct.preparedCall("{call hy\_pro1(?,?)}");

cs.setInt(1,10);

cs.registerOutParameter(2,oracle.jabc.OracleTypes.CURSOR;

cs.execute();

//得到结果集

ResultSet rs=(ResultSet)cs.getObject(2);

while(rs.next()){

System.out.println(rs.getInt(1)+rs.getString(2));

}

【编写分页过程】

有了上面的基础，相信大家可以完成分页存储过程了。

要求，请大家编写一个存储过程，要求可以输入表名、每页显示记录数、当前页。返回总记录数，总页数，和返回的结果集。

如果忘了Oracle中如何分页，请参考第三天的内容。

|  |
| --- |
| *--Oracle的分页,可以把下面的sql语句看做一个模板使用*  select\*from(select t1.\*,rownum rn from(select\*from emp) t1 whererownum<=10)where rn>=6;  *--开发一个包*  createorreplacepackage testpackage as  type test\_cursor isrefcursor;  end testpackage;  /  *--开始编写分页的过程*  createorreplaceprocedure fenye(  tableName invarchar2,  Pagesizeinnumber,*--一页显示的记录数*  pageNow innumber,  myrows outnumber,*--总记录数*  myPageCount outnumber,*--总页数*  p\_cursor out testpackage.test\_cursor*--返回的记录数*  )is  *--定义部分*  *--定义一个sql语句 字符串*  v\_sql varchar2(1000);  *--定义两个整数*  v\_begin number:=(pageNow-1)\*Pagesize+1;  v\_end number:=pageNow+Pagesize;  begin  *--执行部分*  v\_sql:='select \* from (select t1.\*,rownum rn from (select \* from '|| tableName ||') t1 where rownum<='|| v\_end ||') where rn>='||v\_begin;  *--把游标和sql关联*  open p\_cursor for v\_sql;  *--计算myrows和myPageCount*  *--组织一个sql语句*  v\_sql:='select count(\*) from'||tableName;  *--执行sql,并把返回的值,赋给myrows;*  executeimmediate v\_sql into myrows;  *--计算myPageCount*  ifmod(myrows,Pagesize)=0then  myPageCount:=myrows/Pagesize;  else  myPageCount:=myrows/Pagesize+1;  endif;  *--关闭游标*  *--close p\_cursor;*  end;  /  *--Java测试代码*  main(...){  //连接数据库  CallableStatement cs=ct.prepareCall("{call fenye(?,?,?,?,?,?)}");  //给?赋值  cs.setSring(1,"emp");  cs.setInt(2,5);  cs.setInt(3,1);  //注册总记录数  cs.registerOutParameter(4,oracle.jdbc.OracleTypes.INTEGER);  //注册总页数  cs.registerOutParameter(5,oracle.jdbc.OracleTypes.INTEGER);  //注册返回的结果集  cs.registerOutParameter(6,oracle.jdbc.OracleTypes.CURSOR);  cs.execute();  //取出总记录数,这里注意,getInt(4) 中4使用该参数的位置决定的  introwNum=cs.getInt(4);  int pageCount=cs.getInt(5);  ResultSet rs=(resultSet)cs.getObject(6);  //显示一下  System.out.println("rowNum="+rowNum);  System.out.println("总页数="+pageCount);  while(rs.next()){  System.out.println("编号:"+rs.getInt(1)+"名字:"+getString(2));  }  }  *--新的需求,按照薪水从低到高排序,然后取出第6~10的人*  orderby v\_sal |

【例外的处理】

【例外的分类】

Oracle将例外分为预定义例外，非预定义例外和自定义例外三种。

预定义例外用于处理常见的Oracle错误。

非预定义例外用于处理预定义例外不能处理的例外。

自定义例外用于处理与Oracle错误无关的其他情况。

【例外处理】

【例外传递】

如果不处理例外我们看看会出现什么情况：

案例，编写一个过程，可接收雇员的编号，并显示该雇员的姓名。

问题是，如果输入的雇员编号不存在，怎样去处理呢？

|  |
| --- |
| --例外案例  declare  --定义  v\_ename emp.ename%type;  begin  --执行  select ename into v\_ename from emp where empno=&no;  dbms\_output.put\_line('名字:'||v\_ename);  exception  when no\_data\_found then  dbms\_output.put\_line('编号不存在!');  end;  / |

【处理预定义例外】

预定义例外是由pl/sql所提供的系统例外。当pl/sql应用程序违反了Oracle规定的限制时，则会隐含的触发一个内部例外。pl/sql为开发人员提供了20多个预定义例外。我们给大家介绍常用的例外。

【预定义例外】-- case\_not\_found

在开发pl/sql块中编写case语句时，如果在when子句中没有包含必须的条件分支，就会触发case\_not\_found的例外：

create or replace procedure hy\_pro1（hyno number） is

begin

select sal into v\_sal from emp where empno=spno;

case

when v\_sal<1000 then

update emp set sal=sal+100 where empno=hyno;

when v\_sal<2000 then

update emp set sal=sal+200 where empno=hyno;

end case;

exception

when case\_not\_found then

dmbs\_output.put\_line(‘case语句没有与’||v\_sal||’相匹配的条件’);

end;

/

【预定义例外】-- cursor\_already\_open

当重新打开已经打开的游标时，会隐含的触发例外

cursor\_already\_open

declare

cursor emp\_cursor is select ename,sal from emp;

begin

open emp\_cursor;

for emp\_record1 in emp\_cursor loop

dbms\_output.put\_line(emp\_record1.ename);

end loop;

exception

when cursor\_already\_open then

dbms\_output.put\_line(‘游标已经打开’);

end;

/

【预定义例外】-- dup\_val\_on\_index

在唯一索引所对应的列上插入重复的值时，会隐含的触发例外dup\_val\_on\_index例外

begin

insert into dept values（10，‘公关部’，‘北京’）；

exception

when dup\_val\_on\_index then

dbms\_output.put\_line(‘在deptno列上不能出现重复值’)；

end；

/

【预定义例外】-- invalid\_cursor

当试图在不合法的游标上执行操作时，会触发该例外

例如：试图从没有打开的游标提取数据，或是关闭没有打开的游标。则会触发该例外

declare

cursor emp\_cursor is select ename,sal from emp;

emp\_record emp\_cursor%rowtype;

begin

--open emp\_cursor;--打开游标

fetch emp\_cursor into emp\_record;

dbms\_output.put\_line(emp\_record.ename);

close emp\_cursor;

exception

when invalid\_cursor then

dbms\_output.put\_line(‘请检测游标是否打开’);

end;

/

【预定义例外】-- invalid\_number

当输入的数据有误时，会触发该例外

比如：数字100写成loo就会触发该例外

begin

update emp set sal=sal+’loo’;

exception

when invalid\_number then

dbms\_output.put\_line(‘输入的数字不正确’);

end;

/

【预定义例外】-- no\_data\_found

下面是一个pl/sql块，当执行select into 没有返回行，就会触发该例外

declare

v\_sal emp.sal%type;

begin

select sal into v\_sal from emp where ename=’&name’;

exception

when no\_data\_found then

dbms\_output.put\_line(‘不存在该员工’);

end;

/

【预定义例外】-- too\_many\_rows

当执行select into语句时，如果返回超过了一行，则会触发该例外。

declare

v-ename emp.ename%type;

begin

select ename into v\_ename from emp;

exception

when too\_many+rows then

dbms\_output.put\_line(‘返回了多行’);

end;

/

【预定义例外】-- zero\_divide

当执行2/0 语句时，如果返回超过了一行，则会触发该例外。

declare

v\_ename emp.ename%type;

begin

select ename into v\_ename from emp;

exception

when too\_many\_rows then

dbms\_output.put\_line(‘’);

den;

/

【预定义例外】-- value\_error

当执行赋值操作时，如果变量的长度不足以容纳实际数据，则会触发该例外value\_error，比如：

declare

v\_ename varchar2（5）；

begin

selectename into v\_ename from emp where empno=&no;

dbms\_output.put\_line(‘变量尺寸不足’);

end;

/

【其他预定义例外】

1. login\_denide

当用户非法登录时，会触发该例外

1. not\_logged\_no

如果用户没有登录就执行dml操作，就会触发该例外

1. storage\_error

如果超出了内存空间或是内存被损坏，就触发该例外

1. timeout\_on\_resource

如果Oracle在等待资源时，出现了超时就触发该例外

【非预定义例外】

非预定义例外用于处理与例外无关的Oracle错误。使用预定义例外只能处理21个Oracle错误。比如在pl/sql开发应用程序时，可能会遇到其他的一些Oracle错误。比如在pl/sql块中执行dml语句时，违反了约束规定等等，在这样的情况下，也可以处理Oracle的各种例外，因为非预定义例外用的不多，这里我就不举例了。

【处理自定义例外】

预定义例外和自定义例外都是与Oracle错误相关的，并且出现的Oracle错误会隐含的触发相应的例外；而自定义例外与Oracle错误没有任何关联，它是由开发人员为特定情况所定义的例外。

？请编写一个pl/sql块，接收一个雇员的编号，并给该雇员工资增加1000元，如果该雇员不存在，请提示。

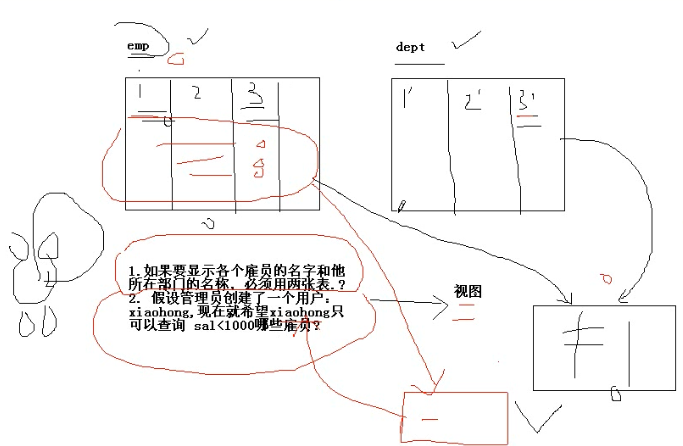
|  |
| --- |
| --自定义例外  create or replace procedure ex\_test(hyNo number) is  --定义一个例外  myex exception;  begin  --更新用户sal  update emp set sal=sal+1000 where empno=hyNo;  --sql%notfound这是表示没有update  --raise myex;触发myex;  if sql%notfound then  raise myex;  end if;  exception  when myex then  dbms\_output.put\_line('没有更新任何用户');  end;  / |

【Oracle视图】

【介绍】

视图是一个虚拟表，其内容由查询定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。

画图说明：



【视图与表的区别】

1. 表需要占用磁盘空间，视图不需要
2. 视图不能添加索引
3. 使用视图可以简化复杂查询。比如：学生选课系统
4. 视图有利于提高安全性

比如：不同用户查看不同视图。

【创建视图】

create view 视图名 as select语句 [with read only]

|  |
| --- |
| *--创建视图,把emp表的薪水<1000的雇员映射到该视图(view)*  Create view myview as select\*from emp where sal<1000;  *--为了简化操作,用一个视图解决显示雇员编号,姓名和部门名称*  Create view myview2 as select emp.empno,emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno;  *--视图之间也可以联合查询* |

【创建或修改视图】

create or replace view 视图名 as select语句 [with read only]

【删除视图】

drop view 视图名

\*\*当表结构过于复杂,请使用视图吧!!!

***第七部分 Oracle SQL性能优化系列***

**标题 ORACLE SQL性能优化系列 black\_snail（翻译）**

**关键字 ORACEL SQL Performance tuning**

**出处 http://www.dbasupport.com/**

**ORACLE SQL性能优化系列**

[1. 选用适合的ORACLE优化器 2](#_Toc50809198)

[2.       访问Table的方式 3](#_Toc50809199)

[3.       共享SQL语句 3](#_Toc50809200)

[4. 选择最有效率的表名顺序(只在基于规则的优化器中有效) 6](#_Toc50809201)

[5.        WHERE子句中的连接顺序． 7](#_Toc50809202)

[6.     SELECT子句中避免使用 ‘ \* ‘ 8](#_Toc50809203)

[7.      减少访问数据库的次数 9](#_Toc50809204)

[8.       使用DECODE函数来减少处理时间 10](#_Toc50809205)

[9.       整合简单,无关联的数据库访问 11](#_Toc50809206)

[10.       删除重复记录 13](#_Toc50809207)

[11.       用TRUNCATE替代DELETE 13](#_Toc50809208)

[12.       尽量多使用COMMIT 13](#_Toc50809209)

[13.       计算记录条数 14](#_Toc50809210)

[14.       用Where子句替换HAVING子句 14](#_Toc50809211)

[15.       减少对表的查询 15](#_Toc50809212)

[16. 通过内部函数提高SQL效率 16](#_Toc50809213)

[17.       使用表的别名(Alias) 18](#_Toc50809214)

[18.       用EXISTS替代IN 18](#_Toc50809215)

[19.       用NOT EXISTS替代NOT IN 19](#_Toc50809216)

[20.       用表连接替换EXISTS 20](#_Toc50809217)

[21.       用EXISTS替换DISTINCT 21](#_Toc50809218)

[22.       识别’低效执行’的SQL语句 22](#_Toc50809219)

[23.       使用TKPROF 工具来查询SQL性能状态 22](#_Toc50809220)

[24.       用EXPLAIN PLAN 分析SQL语句 23](#_Toc50809221)

[25.       用索引提高效率 25](#_Toc50809222)

[26.       索引的操作 26](#_Toc50809223)

[27.       基础表的选择 28](#_Toc50809224)

[28.       多个平等的索引 29](#_Toc50809225)

[29.       等式比较和范围比较 29](#_Toc50809226)

[30.       不明确的索引等级 30](#_Toc50809227)

[31.       强制索引失效 32](#_Toc50809228)

[32. 避免在索引列上使用计算． 33](#_Toc50809229)

[33.       自动选择索引 34](#_Toc50809230)

[34.       避免在索引列上使用NOT 34](#_Toc50809231)

[35.       用>=替代> 36](#_Toc50809232)

[36.       用UNION替换OR (适用于索引列) 37](#_Toc50809233)

[37.       用IN来替换OR 41](#_Toc50809234)

[38.       避免在索引列上使用IS NULL和IS NOT NULL 42](#_Toc50809235)

[39.       总是使用索引的第一个列 43](#_Toc50809236)

[40.       ORACLE内部操作 44](#_Toc50809237)

[41.       用UNION-ALL 替换UNION ( 如果有可能的话) 44](#_Toc50809238)

[42.       使用提示(Hints) 46](#_Toc50809239)

[43.       用WHERE替代ORDER BY 47](#_Toc50809240)

[44.       避免改变索引列的类型. 49](#_Toc50809241)

[45.       需要当心的WHERE子句 50](#_Toc50809242)

[46.       连接多个扫描 52](#_Toc50809243)

[47.       CBO下使用更具选择性的索引 53](#_Toc50809244)

[48.       避免使用耗费资源的操作 54](#_Toc50809245)

[49.       优化GROUP BY 55](#_Toc50809246)

[50.       使用日期 55](#_Toc50809247)

[51.   使用显式的游标(CURSORs) 56](#_Toc50809248)

[52.   优化EXPORT和IMPORT 56](#_Toc50809249)

[53.   分离表和索引 57](#_Toc50809250)

# 1. 选用适合的ORACLE优化器

ORACLE的优化器共有3种:

a. RULE (基于规则) b. COST (基于成本) c. CHOOSE (选择性)

设置缺省的优化器,可以通过对init.ora文件中OPTIMIZER\_MODE参数的各种声明,如RULE,COST,CHOOSE,ALL\_ROWS,FIRST\_ROWS . 你当然也在SQL句级或是会话(session)级对其进行覆盖.

为了使用基于成本的优化器(CBO, Cost-Based Optimizer) , 你必须经常运行analyze 命令,以增加数据库中的对象统计信息(object statistics)的准确性.

如果数据库的优化器模式设置为选择性(CHOOSE),那么实际的优化器模式将和是否运行过analyze命令有关. 如果table已经被analyze过, 优化器模式将自动成为CBO , 反之,数据库将采用RULE形式的优化器.

在缺省情况下,ORACLE采用CHOOSE优化器, 为了避免那些不必要的全表扫描(full table scan) , 你必须尽量避免使用CHOOSE优化器,而直接采用基于规则或者基于成本的优化器.

# 2.       访问Table的方式

ORACLE 采用两种访问表中记录的方式:

a.       全表扫描

全表扫描就是顺序地访问表中每条记录. ORACLE采用一次读入多个数据块(database block)的方式优化全表扫描.

b.       通过ROWID访问表

你可以采用基于ROWID的访问方式情况,提高访问表的效率, , ROWID包含了表中记录的物理位置信息..ORACLE采用索引(INDEX)实现了数据和存放数据的物理位置(ROWID)之间的联系. 通常索引提供了快速访问ROWID的方法,因此那些基于索引列的查询就可以得到性能上的提高.

# 3.       共享SQL语句

为了不重复解析相同的SQL语句,在第一次解析之后, ORACLE将SQL语句存放在内存中.这块位于系统全局区域SGA(system global area)的共享池(shared buffer pool)中的内存可以被所有的数据库用户共享. 因此,当你执行一个SQL语句(有时被称为一个游标)时,如果它

和之前的执行过的语句完全相同, ORACLE就能很快获得已经被解析的语句以及最好的

执行路径. ORACLE的这个功能大大地提高了SQL的执行性能并节省了内存的使用.

可惜的是ORACLE只对简单的表提供高速缓冲(cache buffering) ,这个功能并不适用于多表连接查询.

数据库管理员必须在init.ora中为这个区域设置合适的参数,当这个内存区域越大,就可以保留更多的语句,当然被共享的可能性也就越大了.

当你向ORACLE 提交一个SQL语句,ORACLE会首先在这块内存中查找相同的语句.

这里需要注明的是,ORACLE对两者采取的是一种严格匹配,要达成共享,SQL语句必须

完全相同(包括空格,换行等).

共享的语句必须满足三个条件:

A.      字符级的比较:

当前被执行的语句和共享池中的语句必须完全相同.

例如:

SELECT \* FROM EMP;

和下列每一个都不同

SELECT \* from EMP;

Select \* From Emp;

SELECT \* FROM EMP;

B.      两个语句所指的对象必须完全相同:

例如:

用户 对象名 如何访问

Jack sal\_limit private synonym

Work\_city public synonym

Plant\_detail public synonym

Jill sal\_limit private synonym

Work\_city public synonym

Plant\_detail table owner

考虑一下下列SQL语句能否在这两个用户之间共享.

|  |  |  |
| --- | --- | --- |
| **SQL** | **能否共享** | **原因** |
| select max(sal\_cap) from sal\_limit; | 不能 | 每个用户都有一个private synonym - sal\_limit , 它们是不同的对象 |
| select count(\*0 from work\_city where sdesc like 'NEW%'; | 能 | 两个用户访问相同的对象public synonym - work\_city |
| select a.sdesc,b.location from work\_city a , plant\_detail b where a.city\_id = b.city\_id | 不能 | 用户jack 通过private synonym访问plant\_detail 而jill 是表的所有者,对象不同. |

C.      两个SQL语句中必须使用相同的名字的绑定变量(bind variables)

例如：

第一组的两个SQL语句是相同的(可以共享),而第二组中的两个语句是不同的(即使在运行时,赋于不同的绑定变量相同的值)

a.

select pin , name from people where pin = :blk1.pin;

select pin , name from people where pin = :blk1.pin;

b.

select pin , name from people where pin = **:blk1.ot\_ind**;

select pin , name from people where pin = **:blk1.ov\_ind**;

# 4. 选择最有效率的表名顺序(只在基于规则的优化器中有效)

**ORACLE的解析器按照从右到左的顺序处理FROM子句中的表名**,因此FROM子句中写在最后的表(基础表 driving table)将被最先处理. 在FROM子句中包含多个表的情况下,你必须选择记录条数最少的表作为基础表.当ORACLE处理多个表时, 会运用排序及合并的方式连接它们.首先,扫描第一个表(FROM子句中最后的那个表)并对记录进行派序,然后扫描第二个表(FROM子句中最后第二个表),最后将所有从第二个表中检索出的记录与第一个表中合适记录进行合并.

例如:

表 TAB1 16,384 条记录

表 TAB2 1 条记录

选择TAB2作为基础表 (最好的方法)

select count(\*) from tab1,tab2 执行时间0.96秒

选择TAB2作为基础表 (不佳的方法)

select count(\*) from tab2,tab1 执行时间26.09秒

如果有3个以上的表连接查询, 那就需要选择交叉表(intersection table)作为基础表, 交叉表是指那个被其他表所引用的表.

例如:

EMP表描述了LOCATION表和CATEGORY表的交集.

SELECT \*

FROM LOCATION L ,

CATEGORY C,

**EMP E**

WHERE E.EMP\_NO BETWEEN 1000 AND 2000

AND E.CAT\_NO = C.CAT\_NO

AND E.LOCN = L.LOCN

将比下列SQL更有效率

SELECT \*

FROM **EMP E ,**

LOCATION L ,

CATEGORY C

WHERE E.CAT\_NO = C.CAT\_NO

AND E.LOCN = L.LOCN

AND E.EMP\_NO BETWEEN 1000 AND 2000

# **5.**        WHERE子句中的连接顺序．

**ORACLE采用自下而上的顺序解析WHERE子句**,根据这个原理,表之间的连接必须写在其他WHERE条件之前, 那些可以过滤掉最大数量记录的条件必须写在WHERE子句的末尾.尽可能在where的尾部避免子查询

例如:

(低效,执行时间156.3秒)

SELECT …

FROM EMP E

WHERE **SAL > 50000**

**AND JOB = ‘MANAGER’**

AND 25 < (SELECT COUNT(\*) FROM EMP

WHERE MGR=E.EMPNO);

(高效,执行时间10.6秒)

SELECT …

FROM EMP E

WHERE 25 < (SELECT COUNT(\*) FROM EMP

WHERE MGR=E.EMPNO)

**AND SAL > 50000**

**AND JOB = ‘MANAGER’;**

# 6.     SELECT子句中避免使用 ‘ \* ‘

当你想在SELECT子句中列出所有的COLUMN时,使用动态SQL列引用 ‘\*’ 是一个方便的方法.不幸的是,这是一个非常低效的方法. 实际上,ORACLE在解析的过程中, 会将’\*’ 依次转换成所有的列名, 这个工作是通过查询数据字典完成的, 这意味着将耗费更多的时间.

# 7.      减少访问数据库的次数

当执行每条SQL语句时, ORACLE在内部执行了许多工作: 解析SQL语句, 估算索引的利用率, 绑定变量 , 读数据块等等. 由此可见, 减少访问数据库的次数 , 就能实际上减少ORACLE的工作量.

例如,

以下有三种方法可以检索出雇员号等于0342或0291的职员.

方法1 (最低效)

SELECT EMP\_NAME , SALARY , GRADE

**FROM EMP**

WHERE EMP\_NO = 342;

SELECT EMP\_NAME , SALARY , GRADE

**FROM EMP**

WHERE EMP\_NO = 291;

方法2 (次低效)

DECLARE

CURSOR C1 (E\_NO NUMBER) IS

SELECT EMP\_NAME,SALARY,GRADE

FROM EMP

WHERE EMP\_NO = E\_NO;

BEGIN

OPEN C1(342);

**FETCH C1 INTO …,..,.. ;**

…..

OPEN C1(291);

**FETCH C1 INTO …,..,.. ;**

CLOSE C1;

END;

方法3 (高效)

SELECT A.EMP\_NAME , A.SALARY , A.GRADE,

B.EMP\_NAME , B.SALARY , B.GRADE

**FROM EMP A,EMP B**

WHERE A.EMP\_NO = 342

AND B.EMP\_NO = 291;

**注意:**

在SQL\*Plus , SQL\*Forms和Pro\*C中重新设置ARRAYSIZE参数, 可以增加每次数据库访问的检索数据量 ,建议值为200

(待续)

# 8.       使用DECODE函数来减少处理时间

使用DECODE函数可以避免重复扫描相同记录或重复连接相同的表.

例如:

SELECT COUNT(\*)，SUM(SAL)

FROM　EMP

WHERE DEPT\_NO = 0020

AND ENAME LIKE　‘SMITH%’;

SELECT COUNT(\*)，SUM(SAL)

FROM　EMP

WHERE DEPT\_NO = 0030

AND ENAME LIKE　‘SMITH%’;

你可以用DECODE函数高效地得到相同结果

SELECT COUNT(DECODE(DEPT\_NO,0020,’X’,NULL)) D0020\_COUNT,

COUNT(DECODE(DEPT\_NO,0030,’X’,NULL)) D0030\_COUNT,

SUM(DECODE(DEPT\_NO,0020,SAL,NULL)) D0020\_SAL,

SUM(DECODE(DEPT\_NO,0030,SAL,NULL)) D0030\_SAL

FROM EMP WHERE ENAME LIKE ‘SMITH%’;

类似的,DECODE函数也可以运用于GROUP BY 和ORDER BY子句中.

# 9.       整合简单,无关联的数据库访问

如果你有几个简单的数据库查询语句,你可以把它们整合到一个查询中(即使它们之间没有关系)

例如:

SELECT NAME

FROM EMP

WHERE EMP\_NO = 1234;

SELECT NAME

FROM DPT

WHERE DPT\_NO = 10 ;

SELECT NAME

FROM CAT

WHERE CAT\_TYPE = ‘RD’;

上面的3个查询可以被合并成一个:

SELECT E.NAME , D.NAME , C.NAME

FROM CAT C , DPT D , EMP E,DUAL X

WHERE NVL(‘X’,X.DUMMY) = NVL(‘X’,E.ROWID(+))

AND NVL(‘X’,X.DUMMY) = NVL(‘X’,D.ROWID(+))

AND NVL(‘X’,X.DUMMY) = NVL(‘X’,C.ROWID(+))

AND E.EMP\_NO(+) = 1234

AND D.DEPT\_NO(+) = 10

AND C.CAT\_TYPE(+) = ‘RD’;

**(译者按: 虽然采取这种方法,效率得到提高,但是程序的可读性大大降低,所以读者 还是要权衡之间的利弊)**

# 10.       删除重复记录

最高效的删除重复记录方法 ( 因为使用了ROWID)

DELETE FROM EMP E

WHERE E.ROWID > (SELECT MIN(X.ROWID)

FROM EMP X

WHERE X.EMP\_NO = E.EMP\_NO);

# 11.       用TRUNCATE替代DELETE

当删除表中的记录时,在通常情况下, 回滚段(rollback segments ) 用来存放可以被恢复的信息. 如果你没有COMMIT事务,ORACLE会将数据恢复到删除之前的状态(准确地说是

恢复到执行删除命令之前的状况)

而当运用TRUNCATE时, 回滚段不再存放任何可被恢复的信息.当命令运行后,数据不能被恢复.因此很少的资源被调用,执行时间也会很短.

**(译者按: TRUNCATE只在删除全表适用,TRUNCATE是DDL不是DML)**

# 12.       尽量多使用COMMIT

只要有可能,在程序中尽量多使用COMMIT, 这样程序的性能得到提高,需求也会因为COMMIT所释放的资源而减少:

COMMIT所释放的资源:

a.       回滚段上用于恢复数据的信息.

b.       被程序语句获得的锁

c.       redo log buffer 中的空间

d.       ORACLE为管理上述3种资源中的内部花费

**(译者按: 在使用COMMIT时必须要注意到事务的完整性,现实中效率和事务完整性往往是鱼和熊掌不可得兼)**

# 13.       计算记录条数

和一般的观点相反, count(\*) 比count(1)稍快 , 当然如果可以通过索引检索,对索引列的计数仍旧是最快的. 例如 COUNT(EMPNO)

**(译者按: 在CSDN论坛中,曾经对此有过相当热烈的讨论, 作者的观点并不十分准确,通过实际的测试,上述三种方法并没有显著的性能差别)**

# 14.       用Where子句替换HAVING子句

避免使用HAVING子句, HAVING 只会在检索出所有记录之后才对结果集进行过滤. 这个处理需要排序,总计等操作. 如果能通过WHERE子句限制记录的数目,那就能减少这方面的开销.

例如:

低效:

SELECT REGION，AVG(LOG\_SIZE)

FROM LOCATION

GROUP BY REGION

HAVING REGION REGION != ‘SYDNEY’

AND REGION != ‘PERTH’

高效

SELECT REGION，AVG(LOG\_SIZE)

FROM LOCATION

**WHERE REGION REGION != ‘SYDNEY’**

**AND REGION != ‘PERTH’**

GROUP BY REGION

**(译者按: HAVING 中的条件一般用于对一些集合函数的比较,如COUNT() 等等. 除此而外,一般的条件应该写在WHERE子句中)**

# 15.       减少对表的查询

在含有子查询的SQL语句中,要特别注意减少对表的查询.

例如:

低效

SELECT TAB\_NAME

FROM TABLES

WHERE TAB\_NAME = ( SELECT TAB\_NAME

FROM TAB\_COLUMNS

WHERE VERSION = 604)

AND　DB\_VER= ( SELECT DB\_VER

FROM TAB\_COLUMNS

WHERE VERSION = 604)

高效

SELECT TAB\_NAME

FROM TABLES

WHERE (TAB\_NAME,DB\_VER)

= ( SELECT TAB\_NAME,DB\_VER)

FROM TAB\_COLUMNS

WHERE VERSION = 604)

Update 多个Column 例子:

低效:

UPDATE EMP

SET EMP\_CAT = (SELECT MAX(CATEGORY) FROM EMP\_CATEGORIES),

SAL\_RANGE = (SELECT MAX(SAL\_RANGE) FROM EMP\_CATEGORIES)

WHERE EMP\_DEPT = 0020;

高效:

UPDATE EMP

SET (EMP\_CAT, SAL\_RANGE)

= (SELECT MAX(CATEGORY) , MAX(SAL\_RANGE)

FROM EMP\_CATEGORIES)

WHERE EMP\_DEPT = 0020;

# ****16. 通过内部函数提高SQL效率****

SELECT H.EMPNO,E.ENAME,H.HIST\_TYPE,T.TYPE\_DESC,COUNT(\*)

FROM HISTORY\_TYPE T,EMP E,EMP\_HISTORY H

WHERE H.EMPNO = E.EMPNO

AND H.HIST\_TYPE = T.HIST\_TYPE

GROUP BY H.EMPNO,E.ENAME,H.HIST\_TYPE,T.TYPE\_DESC;

通过调用下面的函数可以提高效率.

FUNCTION LOOKUP\_HIST\_TYPE(TYP IN NUMBER) RETURN VARCHAR2

AS

TDESC VARCHAR2(30);

CURSOR C1 IS

SELECT TYPE\_DESC

FROM HISTORY\_TYPE

WHERE HIST\_TYPE = TYP;

BEGIN

OPEN C1;

FETCH C1 INTO TDESC;

CLOSE C1;

RETURN (NVL(TDESC,’?’));

END;

FUNCTION LOOKUP\_EMP(EMP IN NUMBER) RETURN VARCHAR2

AS

ENAME VARCHAR2(30);

CURSOR C1 IS

SELECT ENAME

FROM EMP

WHERE EMPNO=EMP;

BEGIN

OPEN C1;

FETCH C1 INTO ENAME;

CLOSE C1;

RETURN (NVL(ENAME,’?’));

END;

SELECT H.EMPNO,**LOOKUP\_EMP(H.EMPNO),**

H.HIST\_TYPE,**LOOKUP\_HIST\_TYPE(H.HIST\_TYPE)**,COUNT(\*)

FROM EMP\_HISTORY H

GROUP BY H.EMPNO , H.HIST\_TYPE;

**(译者按: 经常在论坛中看到如 ’能不能用一个SQL写出….’ 的贴子, 殊不知复杂的SQL往往牺牲了执行效率. 能够掌握上面的运用函数解决问题的方法在实际工作中是非常有意义的)**

 (待续)

# 17.       使用表的别名(Alias)

当在SQL语句中连接多个表时, 请使用表的别名并把别名前缀于每个Column上.这样一来,就可以减少解析的时间并减少那些由Column歧义引起的语法错误.

**(译者注: Column歧义指的是由于SQL中不同的表具有相同的Column名,当SQL语句中出现这个Column时,SQL解析器无法判断这个Column的归属)**

# 18.       用EXISTS替代IN

在许多基于基础表的查询中,为了满足一个条件,往往需要对另一个表进行联接.在这种情况下, 使用EXISTS(或NOT EXISTS)通常将提高查询的效率.

低效:

SELECT \*

FROM EMP (基础表)

WHERE EMPNO > 0

AND DEPTNO IN (SELECT DEPTNO

FROM DEPT

WHERE LOC = ‘MELB’)

高效:

SELECT \*

FROM EMP (基础表)

WHERE EMPNO > 0

AND EXISTS (SELECT ‘X’

FROM DEPT

WHERE DEPT.DEPTNO = EMP.DEPTNO

AND LOC = ‘MELB’)

**(译者按: 相对来说,用NOT EXISTS替换NOT IN 将更显著地提高效率,下一节中将指出)**

# 19.       用NOT EXISTS替代NOT IN

在子查询中,NOT IN子句将执行一个内部的排序和合并. 无论在哪种情况下,NOT IN都是最低效的 (因为它对子查询中的表执行了一个全表遍历). 为了避免使用NOT IN ,我们可以把它改写成外连接(Outer Joins)或NOT EXISTS.

例如:

SELECT …

FROM EMP

WHERE DEPT\_NO NOT IN (SELECT DEPT\_NO

FROM DEPT

WHERE DEPT\_CAT=’A’);

为了提高效率.改写为:

(方法一: 高效)

SELECT ….

FROM EMP A,DEPT B

WHERE A.DEPT\_NO = B.DEPT(+)

AND B.DEPT\_NO IS NULL

AND B.DEPT\_CAT(+) = ‘A’

(方法二: 最高效)

SELECT ….

FROM EMP E

WHERE NOT EXISTS (SELECT ‘X’

FROM DEPT D

WHERE D.DEPT\_NO = E.DEPT\_NO

AND DEPT\_CAT = ‘A’);

(待续)

# 20.       用表连接替换EXISTS

通常来说 , 采用表连接的方式比EXISTS更有效率

SELECT ENAME

FROM EMP E

WHERE EXISTS (SELECT ‘X’

FROM DEPT

WHERE DEPT\_NO = E.DEPT\_NO

AND DEPT\_CAT = ‘A’);

(更高效)

SELECT ENAME

FROM DEPT D,EMP E

WHERE E.DEPT\_NO = D.DEPT\_NO

AND DEPT\_CAT = ‘A’ ;

**(译者按: 在RBO的情况下,前者的执行路径包括FILTER,后者使用NESTED LOOP)**

# 21.       用EXISTS替换DISTINCT

当提交一个包含一对多表信息(比如部门表和雇员表)的查询时,避免在SELECT子句中使用DISTINCT. 一般可以考虑用EXIST替换

例如:

低效:

SELECT **DISTINCT** DEPT\_NO,DEPT\_NAME

FROM DEPT D,EMP E

WHERE D.DEPT\_NO = E.DEPT\_NO

高效:

SELECT DEPT\_NO,DEPT\_NAME

FROM DEPT D

WHERE EXISTS ( SELECT ‘X’

FROM EMP E

WHERE E.DEPT\_NO = D.DEPT\_NO);

EXISTS 使查询更为迅速,因为RDBMS核心模块将在子查询的条件一旦满足后,立刻返回结果.

# 22.       识别’低效执行’的SQL语句

用下列SQL工具找出低效SQL:

SELECT EXECUTIONS , DISK\_READS, BUFFER\_GETS,

ROUND((BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS,2) Hit\_radio,

ROUND(DISK\_READS/EXECUTIONS,2) Reads\_per\_run,

SQL\_TEXT

FROM V$SQLAREA

WHERE EXECUTIONS>0

AND BUFFER\_GETS > 0

AND (BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS < 0.8

ORDER BY 4 DESC;

**(译者按: 虽然目前各种关于SQL优化的图形化工具层出不穷,但是写出自己的SQL工具来解决问题始终是一个最好的方法)**

# 23.       使用TKPROF 工具来查询SQL性能状态

SQL trace 工具收集正在执行的SQL的性能状态数据并记录到一个跟踪文件中. 这个跟踪文件提供了许多有用的信息,例如解析次数.执行次数,CPU使用时间等.这些数据将可以用来优化你的系统.

设置SQL TRACE在会话级别: 有效

ALTER SESSION SET SQL\_TRACE TRUE

设置SQL TRACE 在整个数据库有效仿, 你必须将SQL\_TRACE参数在init.ora中设为TRUE, USER\_DUMP\_DEST参数说明了生成跟踪文件的目录

**(译者按: 这一节中,作者并没有提到TKPROF的用法, 对SQL TRACE的用法也不够准确, 设置SQL TRACE首先要在init.ora中设定TIMED\_STATISTICS, 这样才能得到那些重要的时间状态. 生成的trace文件是不可读的,所以要用TKPROF工具对其进行转换,TKPROF有许多执行参数. 大家可以参考ORACLE手册来了解具体的配置. )**

# 24.       用EXPLAIN PLAN 分析SQL语句

EXPLAIN PLAN 是一个很好的分析SQL语句的工具,它甚至可以在不执行SQL的情况下分析语句. 通过分析,我们就可以知道ORACLE是怎么样连接表,使用什么方式扫描表(索引扫描或全表扫描)以及使用到的索引名称.

你需要按照从里到外,从上到下的次序解读分析的结果. EXPLAIN PLAN分析的结果是用缩进的格式排列的, 最内部的操作将被最先解读, 如果两个操作处于同一层中,带有最小操作号的将被首先执行.

NESTED LOOP是少数不按照上述规则处理的操作, 正确的执行路径是检查对NESTED LOOP提供数据的操作,其中操作号最小的将被最先处理.

**译者按:**

**通过实践, 感到还是用SQLPLUS中的SET TRACE 功能比较方便.**

**举例:**

**SQL> list**

**1 SELECT \***

**2 FROM dept, emp**

**3\* WHERE emp.deptno = dept.deptno**

**SQL> set autotrace traceonly /\*traceonly 可以不显示执行结果\*/**

**SQL> /**

**14 rows selected.**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 NESTED LOOPS**

**2 1 TABLE ACCESS (FULL) OF 'EMP'**

**3 1 TABLE ACCESS (BY INDEX ROWID) OF 'DEPT'**

**4 3 INDEX (UNIQUE SCAN) OF 'PK\_DEPT' (UNIQUE)**

**Statistics**

**----------------------------------------------------------**

**0 recursive calls**

**2 db block gets**

**30 consistent gets**

**0 physical reads**

**0 redo size**

**2598 bytes sent via SQL\*Net to client**

**503 bytes received via SQL\*Net from client**

**2 SQL\*Net roundtrips to/from client**

**0 sorts (memory)**

**0 sorts (disk)**

**14 rows processed**

**通过以上分析,可以得出实际的执行步骤是:**

**1.       TABLE ACCESS (FULL) OF 'EMP'**

**2.       INDEX (UNIQUE SCAN) OF 'PK\_DEPT' (UNIQUE)**

**3.       TABLE ACCESS (BY INDEX ROWID) OF 'DEPT'**

**4.       NESTED LOOPS (JOINING 1 AND 3)**

**注: 目前许多第三方的工具如TOAD和ORACLE本身提供的工具如OMS的SQL Analyze都提供了极其方便的EXPLAIN PLAN工具.也许喜欢图形化界面的朋友们可以选用它们.**

(待续)

# 25.       用索引提高效率

索引是表的一个概念部分,用来提高检索数据的效率. 实际上,ORACLE使用了一个复杂的自平衡B-tree结构. 通常,通过索引查询数据比全表扫描要快. 当ORACLE找出执行查询和Update语句的最佳路径时, ORACLE优化器将使用索引. 同样在联结多个表时使用索引也可以提高效率. 另一个使用索引的好处是,它提供了主键(primary key)的唯一性验证.

除了那些LONG或LONG RAW数据类型, 你可以索引几乎所有的列. 通常, 在大型表中使用索引特别有效. 当然,你也会发现, 在扫描小表时,使用索引同样能提高效率.

虽然使用索引能得到查询效率的提高,但是我们也必须注意到它的代价. 索引需要空间来

存储,也需要定期维护, 每当有记录在表中增减或索引列被修改时, 索引本身也会被修改. 这意味着每条记录的INSERT , DELETE , UPDATE将为此多付出4 , 5 次的磁盘I/O . 因为索引需要额外的存储空间和处理,那些不必要的索引反而会使查询反应时间变慢.

**译者按:**

**定期的重构索引是有必要的.**

**ALTER INDEX <INDEXNAME> REBUILD <TABLESPACENAME>**

# 26.       索引的操作

ORACLE对索引有两种访问模式.

索引唯一扫描 ( INDEX UNIQUE SCAN)

大多数情况下, 优化器通过WHERE子句访问INDEX.

例如:

表LODGING有两个索引 : 建立在LODGING列上的唯一性索引LODGING\_PK和建立在MANAGER列上的非唯一性索引LODGING$MANAGER.

SELECT \*

FROM LODGING

WHERE LODGING = ‘ROSE HILL’;

在内部 , 上述SQL将被分成两步执行, 首先 , LODGING\_PK 索引将通过索引唯一扫描的方式被访问 , 获得相对应的ROWID, 通过ROWID访问表的方式 执行下一步检索.

如果被检索返回的列包括在INDEX列中,ORACLE将不执行第二步的处理(通过ROWID访问表). 因为检索数据保存在索引中, 单单访问索引就可以完全满足查询结果.

下面SQL只需要INDEX UNIQUE SCAN 操作.

SELECT LODGING

FROM LODGING

WHERE LODGING = ‘ROSE HILL’;

索引范围查询(INDEX RANGE SCAN)

适用于两种情况:

1.       基于一个范围的检索

2.       基于非唯一性索引的检索

例1:

SELECT LODGING

FROM LODGING

WHERE LODGING LIKE ‘M%’;

WHERE子句条件包括一系列值, ORACLE将通过索引范围查询的方式查询LODGING\_PK . 由于索引范围查询将返回一组值, 它的效率就要比索引唯一扫描

低一些.

例2:

SELECT LODGING

FROM LODGING

WHERE MANAGER = ‘BILL GATES’;

这个SQL的执行分两步, LODGING$MANAGER的索引范围查询(得到所有符合条件记录的ROWID) 和下一步同过ROWID访问表得到LODGING列的值. 由于LODGING$MANAGER是一个非唯一性的索引,数据库不能对它执行索引唯一扫描.

由于SQL返回LODGING列,而它并不存在于LODGING$MANAGER索引中, 所以在索引范围查询后会执行一个通过ROWID访问表的操作.

WHERE子句中, 如果索引列所对应的值的第一个字符由通配符(WILDCARD)开始, 索引将不被采用.

SELECT LODGING

FROM LODGING

WHERE MANAGER LIKE ‘％HANMAN’;

在这种情况下，ORACLE将使用全表扫描.

(待续)

# 27.       基础表的选择

基础表(Driving Table)是指被最先访问的表(通常以全表扫描的方式被访问). 根据优化器的不同, SQL语句中基础表的选择是不一样的.

如果你使用的是CBO (COST BASED OPTIMIZER),优化器会检查SQL语句中的每个表的物理大小,索引的状态,然后选用花费最低的执行路径.

如果你用RBO (RULE BASED OPTIMIZER) , 并且所有的连接条件都有索引对应, 在这种情况下, 基础表就是FROM 子句中列在最后的那个表.

举例:

SELECT A.NAME , B.MANAGER

FROM　WORKER A,

LODGING B

WHERE　A.LODGING = B.LODING;

由于LODGING表的LODING列上有一个索引, 而且WORKER表中没有相比较的索引, WORKER表将被作为查询中的基础表.

# 28.       多个平等的索引

当SQL语句的执行路径可以使用分布在多个表上的多个索引时, ORACLE会同时使用多个索引并在运行时对它们的记录进行合并, 检索出仅对全部索引有效的记录.

在ORACLE选择执行路径时,唯一性索引的等级高于非唯一性索引. 然而这个规则只有

当WHERE子句中索引列和常量比较才有效.如果索引列和其他表的索引类相比较. 这种子句在优化器中的等级是非常低的.

如果不同表中两个想同等级的索引将被引用, FROM子句中表的顺序将决定哪个会被率先使用. FROM子句中最后的表的索引将有最高的优先级.

如果相同表中两个想同等级的索引将被引用, WHERE子句中最先被引用的索引将有最高的优先级.

举例:

DEPTNO上有一个非唯一性索引,EMP\_CAT也有一个非唯一性索引.

SELECT ENAME,

FROM EMP

WHERE DEPT\_NO = 20

AND EMP\_CAT = ‘A’;

这里,DEPTNO索引将被最先检索,然后同EMP\_CAT索引检索出的记录进行合并. 执行路径如下:

TABLE ACCESS BY ROWID ON EMP

AND-EQUAL

INDEX RANGE SCAN ON DEPT\_IDX

INDEX RANGE SCAN ON CAT\_IDX

# 29.       等式比较和范围比较

当WHERE子句中有索引列**,** ORACLE不能合并它们,ORACLE将用范围比较.

举例:

DEPTNO上有一个非唯一性索引,EMP\_CAT也有一个非唯一性索引.

SELECT ENAME

FROM EMP

WHERE DEPTNO > 20

AND EMP\_CAT = ‘A’;

这里只有EMP\_CAT索引被用到,然后所有的记录将逐条与DEPTNO条件进行比较. 执行路径如下:

TABLE ACCESS BY ROWID ON EMP

INDEX RANGE SCAN ON CAT\_IDX

# 30.       不明确的索引等级

当ORACLE无法判断索引的等级高低差别,优化器将只使用一个索引,它就是在WHERE子句中被列在最前面的.

举例:

DEPTNO上有一个非唯一性索引,EMP\_CAT也有一个非唯一性索引.

SELECT ENAME

FROM EMP

WHERE DEPTNO > 20

AND EMP\_CAT > ‘A’;

这里, ORACLE只用到了DEPT\_NO索引. 执行路径如下:

TABLE ACCESS BY ROWID ON EMP

INDEX RANGE SCAN ON DEPT\_IDX

**译者按:**

**我们来试一下以下这种情况:**

**SQL> select index\_name, uniqueness from user\_indexes where table\_name = 'EMP';**

**INDEX\_NAME UNIQUENES**

**------------------------------ ---------**

**EMPNO UNIQUE**

**EMPTYPE NONUNIQUE**

**SQL> select \* from emp where empno >= 2 and emp\_type = 'A' ;**

**no rows selected**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (BY INDEX ROWID) OF 'EMP'**

**2 1 INDEX (RANGE SCAN) OF 'EMPTYPE' (NON-UNIQUE)**

**虽然EMPNO是唯一性索引,但是由于它所做的是范围比较, 等级要比非唯一性索引的等式比较低!**

# 31.       强制索引失效

如果两个或以上索引具有相同的等级,你可以强制命令ORACLE优化器使用其中的一个(通过它,检索出的记录数量少) .

举例:

SELECT ENAME

FROM EMP

WHERE EMPNO = 7935

AND DEPTNO + 0 = 10 **/\*DEPTNO上的索引将失效\*/**

AND EMP\_TYPE || ‘’ = ‘A’  **/\*EMP\_TYPE上的索引将失效\*/**

这是一种相当直接的提高查询效率的办法. 但是你必须谨慎考虑这种策略,一般来说,只有在你希望单独优化几个SQL时才能采用它.

这里有一个例子关于何时采用这种策略,

假设在EMP表的EMP\_TYPE列上有一个非唯一性的索引而EMP\_CLASS上没有索引.

SELECT ENAME

FROM EMP

WHERE EMP\_TYPE = ‘A’

AND EMP\_CLASS = ‘X’;

优化器会注意到EMP\_TYPE上的索引并使用它. 这是目前唯一的选择. 如果,一段时间以后, 另一个非唯一性建立在EMP\_CLASS上,优化器必须对两个索引进行选择,在通常情况下,优化器将使用两个索引并在他们的结果集合上执行排序及合并. 然而,如果其中一个索引（EMP\_TYPE）接近于唯一性而另一个索引（EMP\_CLASS）上有几千个重复的值. 排序及合并就会成为一种不必要的负担. 在这种情况下,你希望使优化器屏蔽掉EMP\_CLASS索引.

用下面的方案就可以解决问题.

SELECT ENAME

FROM EMP

WHERE EMP\_TYPE = ‘A’

AND EMP\_CLASS**||’’** = ‘X’;

# 避免在索引列上使用计算．

WHERE子句中，如果索引列是函数的一部分．优化器将不使用索引而使用全表扫描．

举例:

低效：

SELECT …

FROM DEPT

WHERE SAL \* 12 > 25000;

高效:

SELECT …

FROM DEPT

WHERE SAL > 25000/12;

**译者按:**

**这是一个非常实用的规则，请务必牢记**

# 33.       自动选择索引

如果表中有两个以上（包括两个）索引，其中有一个唯一性索引，而其他是非唯一性．

在这种情况下，ORACLE将使用唯一性索引而完全忽略非唯一性索引．

举例:

SELECT ENAME

FROM EMP

WHERE EMPNO = 2326

AND DEPTNO = 20 ;

这里，只有EMPNO上的索引是唯一性的，所以EMPNO索引将用来检索记录．

TABLE ACCESS BY ROWID ON EMP

INDEX UNIQUE SCAN ON EMP\_NO\_IDX

# 34.       避免在索引列上使用NOT

通常，　我们要避免在索引列上使用NOT, NOT会产生在和在索引列上使用函数相同的

影响. 当ORACLE”遇到”NOT,他就会停止使用索引转而执行全表扫描.

举例:

低效: (这里,不使用索引)

SELECT …

FROM DEPT

WHERE DEPT\_CODE NOT = 0;

高效: (这里,使用了索引)

SELECT …

FROM DEPT

WHERE DEPT\_CODE > 0;

**需要注意的是,在某些时候, ORACLE优化器会自动将NOT转化成相对应的关系操作符**.

NOT > to <=

NOT >= to <

NOT < to >=

NOT <= to >

**译者按:**

**在这个例子中,作者犯了一些错误. 例子中的低效率SQL是不能被执行的.**

**我做了一些测试:**

**SQL> select \* from emp where NOT empno > 1;**

**no rows selected**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (BY INDEX ROWID) OF 'EMP'**

**2 1 INDEX (RANGE SCAN) OF 'EMPNO' (UNIQUE)**

**SQL> select \* from emp where empno <= 1;**

**no rows selected**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (BY INDEX ROWID) OF 'EMP'**

**2 1 INDEX (RANGE SCAN) OF 'EMPNO' (UNIQUE)**

**两者的效率完全一样，也许这符合作者关于” 在某些时候, ORACLE优化器会自动将NOT转化成相对应的关系操作符” 的观点．**

# 35.       用>=替代>

如果DEPTNO上有一个索引,

高效:

SELECT \*

FROM EMP

WHERE DEPTNO >=4

低效:

SELECT \*

FROM EMP

WHERE DEPTNO >3

两者的区别在于, 前者DBMS将直接跳到第一个DEPT等于4的记录而后者将首先定位到DEPTNO=3的记录并且向前扫描到第一个DEPT大于3的记录.

# 36.       用UNION替换OR (适用于索引列)

通常情况下, 用UNION替换WHERE子句中的OR将会起到较好的效果. 对索引列使用OR将造成全表扫描. 注意, 以上规则只针对多个索引列有效. 如果有column没有被索引, 查询效率可能会因为你没有选择OR而降低.

在下面的例子中, LOC\_ID 和REGION上都建有索引.

高效:

SELECT LOC\_ID , LOC\_DESC , REGION

FROM LOCATION

WHERE LOC\_ID = 10

UNION

SELECT LOC\_ID , LOC\_DESC , REGION

FROM LOCATION

WHERE REGION = “MELBOURNE”

低效:

SELECT LOC\_ID , LOC\_DESC , REGION

FROM LOCATION

WHERE LOC\_ID = 10 OR REGION = “MELBOURNE”

如果你坚持要用OR, 那就需要返回记录最少的索引列写在最前面.

注意:

WHERE KEY1 = 10 (返回最少记录)

OR KEY2 = 20 (返回最多记录)

ORACLE 内部将以上转换为

WHERE KEY1 = 10 AND

((NOT KEY1 = 10) AND KEY2 = 20)

**译者按:**

**下面的测试数据仅供参考: (a = 1003 返回一条记录 , b = 1 返回1003条记录)**

**SQL> select \* from unionvsor /\*1st test\*/**

**2 where a = 1003 or b = 1;**

**1003 rows selected.**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 CONCATENATION**

**2 1 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**3 2 INDEX (RANGE SCAN) OF 'UB' (NON-UNIQUE)**

**4 1 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**5 4 INDEX (RANGE SCAN) OF 'UA' (NON-UNIQUE)**

**Statistics**

**----------------------------------------------------------**

**0 recursive calls**

**0 db block gets**

**144 consistent gets**

**0 physical reads**

**0 redo size**

**63749 bytes sent via SQL\*Net to client**

**7751 bytes received via SQL\*Net from client**

**68 SQL\*Net roundtrips to/from client**

**0 sorts (memory)**

**0 sorts (disk)**

**1003 rows processed**

**SQL> select \* from unionvsor /\*2nd test\*/**

**2 where b = 1 or a = 1003 ;**

**1003 rows selected.**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 CONCATENATION**

**2 1 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**3 2 INDEX (RANGE SCAN) OF 'UA' (NON-UNIQUE)**

**4 1 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**5 4 INDEX (RANGE SCAN) OF 'UB' (NON-UNIQUE)**

**Statistics**

**----------------------------------------------------------**

**0 recursive calls**

**0 db block gets**

**143 consistent gets**

**0 physical reads**

**0 redo size**

**63749 bytes sent via SQL\*Net to client**

**7751 bytes received via SQL\*Net from client**

**68 SQL\*Net roundtrips to/from client**

**0 sorts (memory)**

**0 sorts (disk)**

**1003 rows processed**

**SQL> select \* from unionvsor /\*3rd test\*/**

**2 where a = 1003**

**3 union**

**4 select \* from unionvsor**

**5 where b = 1;**

**1003 rows selected.**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 SORT (UNIQUE)**

**2 1 UNION-ALL**

**3 2 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**4 3 INDEX (RANGE SCAN) OF 'UA' (NON-UNIQUE)**

**5 2 TABLE ACCESS (BY INDEX ROWID) OF 'UNIONVSOR'**

**6 5 INDEX (RANGE SCAN) OF 'UB' (NON-UNIQUE)**

**Statistics**

**----------------------------------------------------------**

**0 recursive calls**

**0 db block gets**

**10 consistent gets**

**0 physical reads**

**0 redo size**

**63735 bytes sent via SQL\*Net to client**

**7751 bytes received via SQL\*Net from client**

**68 SQL\*Net roundtrips to/from client**

**1 sorts (memory)**

**0 sorts (disk)**

**1003 rows processed**

**用UNION的效果可以从consistent gets和 SQL\*NET的数据交换量的减少看出**

# 37.       用IN来替换OR

下面的查询可以被更有效率的语句替换:

低效:

SELECT….

FROM LOCATION

WHERE LOC\_ID = 10

OR LOC\_ID = 20

OR LOC\_ID = 30

高效

SELECT…

FROM LOCATION

WHERE LOC\_IN IN (10,20,30);

**译者按:**

**这是一条简单易记的规则，但是实际的执行效果还须检验，在ORACLE8i下，两者的执行路径似乎是相同的．**

# 38.       避免在索引列上使用IS NULL和IS NOT NULL

避免在索引中使用任何可以为空的列，ORACLE将无法使用该索引 ．对于单列索引，如果列包含空值，索引中将不存在此记录. 对于复合索引，如果每个列都为空，索引中同样不存在此记录.　如果至少有一个列不为空，则记录存在于索引中．

举例:

如果唯一性索引建立在表的A列和B列上, 并且表中存在一条记录的A,B值为(123,null) , ORACLE将不接受下一条具有相同A,B值（123,null）的记录(插入). 然而如果

所有的索引列都为空，ORACLE将认为整个键值为空而**空不等于空**. 因此你可以插入1000

条具有相同键值的记录,当然它们都是空!

因为空值不存在于索引列中,所以WHERE子句中对索引列进行空值比较将使ORACLE停用该索引.

举例:

低效: (索引失效)

SELECT …

FROM DEPARTMENT

WHERE DEPT\_CODE **IS NOT NULL;**

高效: (索引有效)

SELECT …

FROM DEPARTMENT

WHERE DEPT\_CODE **>=0;**

# 39.       总是使用索引的第一个列

如果索引是建立在多个列上, 只有在它的第一个列(leading column)被where子句引用时,优化器才会选择使用该索引.

**译者按:**

**这也是一条简单而重要的规则. 见以下实例.**

**SQL> create table multiindexusage ( inda number , indb number , descr varchar2(10));**

**Table created.**

**SQL> create index multindex on multiindexusage(inda,indb);**

**Index created.**

**SQL> set autotrace traceonly**

**SQL> select \* from multiindexusage where inda = 1;**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (BY INDEX ROWID) OF 'MULTIINDEXUSAGE'**

**2 1 INDEX (RANGE SCAN) OF 'MULTINDEX' (NON-UNIQUE)**

**SQL> select \* from multiindexusage where indb = 1;**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (FULL) OF 'MULTIINDEXUSAGE'**

**很明显, 当仅引用索引的第二个列时,优化器使用了全表扫描而忽略了索引**

# 40.       ORACLE内部操作

当执行查询时,ORACLE采用了内部的操作. 下表显示了几种重要的内部操作.

|  |  |
| --- | --- |
| ORACLE Clause | 内部操作 |
| ORDER BY | SORT ORDER BY |
| UNION | UNION-ALL |
| MINUS | MINUS |
| INTERSECT | INTERSECT |
| DISTINCT,MINUS,INTERSECT,UNION | SORT UNIQUE |
| MIN,MAX,COUNT | SORT AGGREGATE |
| GROUP BY | SORT GROUP BY |
| ROWNUM | COUNT or COUNT STOPKEY |
| Queries involving Joins | SORT JOIN,MERGE JOIN,NESTED LOOPS |
| CONNECT BY | CONNECT BY |

# 41.       用UNION-ALL 替换UNION ( 如果有可能的话)

当SQL语句需要UNION两个查询结果集合时,这两个结果集合会以UNION-ALL的方式被合并, 然后在输出最终结果前进行排序.

如果用UNION ALL替代UNION, 这样排序就不是必要了. 效率就会因此得到提高.

举例:

低效：

　　　　SELECT ACCT\_NUM, BALANCE\_AMT

FROM DEBIT\_TRANSACTIONS

WHERE TRAN\_DATE = ’31-DEC-95’

UNION

SELECT ACCT\_NUM, BALANCE\_AMT

FROM DEBIT\_TRANSACTIONS

WHERE TRAN\_DATE = ’31-DEC-95’

高效:

SELECT ACCT\_NUM, BALANCE\_AMT

FROM DEBIT\_TRANSACTIONS

WHERE TRAN\_DATE = ’31-DEC-95’

UNION ALL

SELECT ACCT\_NUM, BALANCE\_AMT

FROM DEBIT\_TRANSACTIONS

WHERE TRAN\_DATE = ’31-DEC-95’

**译者按:**

**需要注意的是，UNION ALL 将重复输出两个结果集合中相同记录. 因此各位还是要从业务需求分析使用UNION ALL的可行性.**

**UNION 将对结果集合排序,这个操作会使用到SORT\_AREA\_SIZE这块内存. 对于这块内存的优化也是相当重要的. 下面的SQL可以用来查询排序的消耗量**

**Select substr(name,1,25) "Sort Area Name",**

**substr(value,1,15) "Value"**

**from v$sysstat**

**where name like 'sort%'**

# 42.       使用提示(Hints)

对于表的访问,可以使用两种Hints.

FULL 和 ROWID

FULL hint 告诉ORACLE使用全表扫描的方式访问指定表.

例如:

SELECT /\*+ FULL(EMP) \*/ \*

FROM EMP

WHERE EMPNO = 7893;

ROWID hint 告诉ORACLE使用TABLE ACCESS BY ROWID的操作访问表.

通常, 你需要采用TABLE ACCESS BY ROWID的方式特别是当访问大表的时候, 使用这种方式, 你需要知道ROIWD的值或者使用索引.

如果一个大表没有被设定为缓存(CACHED)表而你希望它的数据在查询结束是仍然停留

在SGA中,你就可以使用CACHE hint 来告诉优化器把数据保留在SGA中. 通常CACHE hint 和 FULL hint 一起使用.

例如:

SELECT /\*+ FULL(WORKER) CACHE(WORKER)\*/ \*

FROM WORK;

索引hint 告诉ORACLE使用基于索引的扫描方式. 你不必说明具体的索引名称

例如:

SELECT /\*+ INDEX(LODGING) \*/ LODGING

FROM LODGING

WHERE MANAGER = ‘BILL GATES’;

在不使用hint的情况下, 以上的查询应该也会使用索引,然而,如果该索引的重复值过多而你的优化器是CBO, 优化器就可能忽略索引. 在这种情况下, 你可以用INDEX hint强制ORACLE使用该索引.

ORACLE hints 还包括ALL\_ROWS, FIRST\_ROWS, RULE,USE\_NL, USE\_MERGE, USE\_HASH 等等.

**译者按:**

**使用hint , 表示我们对ORACLE优化器缺省的执行路径不满意,需要手工修改.这是一个很有技巧性的工作. 我建议只针对特定的,少数的SQL进行hint的优化.对ORACLE的优化器还是要有信心(特别是CBO)**

# 43.       用WHERE替代ORDER BY

ORDER BY 子句只在两种严格的条件下使用索引.

ORDER BY中所有的列必须包含在相同的索引中并保持在索引中的排列顺序.

ORDER BY中所有的列必须定义为非空.

WHERE子句使用的索引和ORDER BY子句中所使用的索引不能并列.

例如:

表DEPT包含以下列:

DEPT\_CODE PK NOT NULL

DEPT\_DESC NOT NULL

DEPT\_TYPE NULL

非唯一性的索引(DEPT\_TYPE)

低效: (索引不被使用)

SELECT DEPT\_CODE

FROM DEPT

ORDER BY DEPT\_TYPE

EXPLAIN PLAN:

SORT ORDER BY

TABLE ACCESS FULL

高效: (使用索引)

SELECT DEPT\_CODE

FROM DEPT

WHERE DEPT\_TYPE > 0

EXPLAIN PLAN:

TABLE ACCESS BY ROWID ON EMP

INDEX RANGE SCAN ON DEPT\_IDX

**译者按:**

**ORDER BY 也能使用索引! 这的确是个容易被忽视的知识点. 我们来验证一下:**

**SQL> select \* from emp order by empno;**

**Execution Plan**

**----------------------------------------------------------**

**0 SELECT STATEMENT Optimizer=CHOOSE**

**1 0 TABLE ACCESS (BY INDEX ROWID) OF 'EMP'**

**2 1 INDEX (FULL SCAN) OF 'EMPNO' (UNIQUE)**

# 44.       避免改变索引列的类型.

当比较不同数据类型的数据时, ORACLE自动对列进行简单的类型转换.

假设 EMPNO是一个数值类型的索引列.

SELECT …

FROM EMP

WHERE EMPNO = ‘123’

实际上,经过ORACLE类型转换, 语句转化为:

SELECT …

FROM EMP

WHERE EMPNO = TO\_NUMBER(‘123’)

幸运的是,类型转换没有发生在索引列上,索引的用途没有被改变.

现在,假设EMP\_TYPE是一个字符类型的索引列.

SELECT …

FROM EMP

WHERE EMP\_TYPE = 123

这个语句被ORACLE转换为:

SELECT …

FROM EMP

WHERE TO\_NUMBER(EMP\_TYPE)=123

因为内部发生的类型转换, 这个索引将不会被用到!

**译者按:**

**为了避免ORACLE对你的SQL进行隐式的类型转换, 最好把类型转换用显式表现出来. 注意当字符和数值比较时, ORACLE会优先转换数值类型到字符类型.**

# 45.       需要当心的WHERE子句

某些SELECT 语句中的WHERE子句不使用索引. 这里有一些例子.

在下面的例子里, ‘!=’ 将不使用索引. 记住, 索引只能告诉你什么存在于表中, 而不能告诉你什么不存在于表中.

不使用索引:

SELECT ACCOUNT\_NAME

FROM TRANSACTION

WHERE AMOUNT !=0;

使用索引:

SELECT ACCOUNT\_NAME

FROM TRANSACTION

WHERE AMOUNT >0;

下面的例子中, ‘||’是字符连接函数. 就象其他函数那样, 停用了索引.

不使用索引:

SELECT ACCOUNT\_NAME,AMOUNT

FROM TRANSACTION

WHERE ACCOUNT\_NAME||ACCOUNT\_TYPE=’AMEXA’;

使用索引:

SELECT ACCOUNT\_NAME,AMOUNT

FROM TRANSACTION

WHERE ACCOUNT\_NAME = ‘AMEX’

AND ACCOUNT\_TYPE=’ A’;

下面的例子中, ‘+’是数学函数. 就象其他数学函数那样, 停用了索引.

不使用索引:

SELECT ACCOUNT\_NAME, AMOUNT

FROM TRANSACTION

WHERE AMOUNT + 3000 >5000;

使用索引:

SELECT ACCOUNT\_NAME, AMOUNT

FROM TRANSACTION

WHERE AMOUNT > 2000 ;

下面的例子中,相同的索引列不能互相比较,这将会启用全表扫描.

不使用索引:

SELECT ACCOUNT\_NAME, AMOUNT

FROM TRANSACTION

WHERE ACCOUNT\_NAME = NVL(:ACC\_NAME,ACCOUNT\_NAME);

使用索引:

SELECT ACCOUNT\_NAME, AMOUNT

FROM TRANSACTION

WHERE ACCOUNT\_NAME LIKE NVL(:ACC\_NAME,’%’);

**译者按:**

**如果一定要对使用函数的列启用索引, ORACLE新的功能: 基于函数的索引(Function-Based Index) 也许是一个较好的方案.**

**CREATE INDEX EMP\_I ON EMP (UPPER(ename)); /\*建立基于函数的索引\*/**

**SELECT \* FROM emp WHERE UPPER(ename) = ‘BLACKSNAIL’; /\*将使用索引\*/**

# 46.       连接多个扫描

如果你对一个列和一组有限的值进行比较, 优化器可能执行多次扫描并对结果进行合并连接.

举例:

SELECT \*

FROM LODGING

WHERE MANAGER IN (‘BILL GATES’,’KEN MULLER’);

优化器可能将它转换成以下形式

SELECT \*

FROM LODGING

WHERE MANAGER = ‘BILL GATES’

OR MANAGER = ’KEN MULLER’;

当选择执行路径时, 优化器可能对每个条件采用LODGING$MANAGER上的索引范围扫描. 返回的ROWID用来访问LODGING表的记录 (通过TABLE ACCESS BY ROWID 的方式). 最后两组记录以连接(CONCATENATION)的形式被组合成一个单一的集合.

Explain Plan :

SELECT STATEMENT Optimizer=CHOOSE

CONCATENATION

TABLE ACCESS (BY INDEX ROWID) OF LODGING

INDEX (RANGE SCAN ) OF LODGING$MANAGER (NON-UNIQUE)

TABLE ACCESS (BY INDEX ROWID) OF LODGING

INDEX (RANGE SCAN ) OF LODGING$MANAGER (NON-UNIQUE)

**译者按:**

**本节和第37节似乎有矛盾之处.**

# 47.       CBO下使用更具选择性的索引

基于成本的优化器(CBO, Cost-Based Optimizer)对索引的选择性进行判断来决定索引的使用是否能提高效率.

如果索引有很高的选择性, 那就是说对于每个不重复的索引键值,只对应数量很少的记录.

比如, 表中共有100条记录而其中有80个不重复的索引键值. 这个索引的选择性就是80/100 = 0.8 . 选择性越高, 通过索引键值检索出的记录就越少.

如果索引的选择性很低, 检索数据就需要大量的索引范围查询操作和ROWID 访问表的

操作. 也许会比全表扫描的效率更低.

**译者按:**

**下列经验请参阅:**

**a.       如果检索数据量超过30%的表中记录数.使用索引将没有显著的效率提高.**

**b.       在特定情况下, 使用索引也许会比全表扫描慢, 但这是同一个数量级上的区别. 而通常情况下,使用索引比全表扫描要块几倍乃至几千倍!**

# 48.       避免使用耗费资源的操作

带有DISTINCT,UNION,MINUS,INTERSECT,ORDER BY的SQL语句会启动SQL引擎

执行耗费资源的排序(SORT)功能. DISTINCT需要一次排序操作, 而其他的至少需要执行两次排序.

例如,一个UNION查询,其中每个查询都带有GROUP BY子句, GROUP BY会触发嵌入排序(NESTED SORT) ; 这样, 每个查询需要执行一次排序, 然后在执行UNION时, 又一个唯一排序(SORT UNIQUE)操作被执行而且它只能在前面的嵌入排序结束后才能开始执行. 嵌入的排序的深度会大大影响查询的效率.

通常, 带有UNION, MINUS , INTERSECT的SQL语句都可以用其他方式重写.

**译者按:**

**如果你的数据库的SORT\_AREA\_SIZE调配得好, 使用UNION , MINUS, INTERSECT也是可以考虑的, 毕竟它们的可读性很强**

# 49.       优化GROUP BY

提高GROUP BY 语句的效率, 可以通过将不需要的记录在GROUP BY 之前过滤掉.下面两个查询返回相同结果但第二个明显就快了许多.

低效:

SELECT JOB , AVG(SAL)

FROM EMP

GROUP JOB

HAVING JOB = ‘PRESIDENT’

OR JOB = ‘MANAGER’

高效:

SELECT JOB , AVG(SAL)

FROM EMP

WHERE JOB = ‘PRESIDENT’

OR JOB = ‘MANAGER’

GROUP JOB

**译者按:**

**本节和14节相同. 可略过.**

# 50.       使用日期

当使用日期是,需要注意如果有超过5位小数加到日期上, 这个日期会进到下一天!

例如:

1.

SELECT TO\_DATE(‘01-JAN-93’+.99999)

FROM DUAL;

Returns:

’01-JAN-93 23:59:59’

2.

SELECT TO\_DATE(‘01-JAN-93’+**.999999**)

FROM DUAL;

Returns:

**’02-JAN-93 00:00:00’**

**译者按:**

**虽然本节和SQL性能优化没有关系, 但是作者的功力可见一斑**

# 51.   使用显式的游标(CURSORs)

使用隐式的游标,将会执行两次操作. 第一次检索记录, 第二次检查TOO MANY ROWS 这个exception . 而显式游标不执行第二次操作.

# 52.   优化EXPORT和IMPORT

使用较大的BUFFER(比如10MB , 10,240,000)可以提高EXPORT和IMPORT的速度.

ORACLE将尽可能地获取你所指定的内存大小,即使在内存不满足,也不会报错.这个值至少要和表中最大的列相当,否则列值会被截断.

**译者按:**

**可以肯定的是, 增加BUFFER会大大提高EXPORT , IMPORT的效率. (曾经碰到过一个CASE, 增加BUFFER后,IMPORT/EXPORT快了10倍!)**

**作者可能犯了一个错误: “这个值至少要和表中最大的列相当,否则列值会被截断. “其中最大的列也许是指最大的记录大小.**

**关于EXPORT/IMPORT的优化,CSDN论坛中有一些总结性的贴子,比如关于BUFFER参数, COMMIT参数等等, 详情请查.**

# 53.   分离表和索引

总是将你的表和索引建立在不同的表空间内(TABLESPACES). 决不要将不属于ORACLE内部系统的对象存放到SYSTEM表空间里. 同时,确保数据表空间和索引表空间置于不同的硬盘上.

**译者按:**

**“同时,确保数据表空间和索引表空间置与不同的硬盘上.”可能改为如下更为准确 “同时,确保数据表空间和索引表空间置与不同的硬盘控制卡控制的硬盘上.”**

**(全文完)**

**Black\_Snail**

[**ligang1000@hotmail.com**](mailto:ligang1000@hotmail.com)

**4/Sep/2003**

***第八部分 索引***

# 索引

[索引 1](#_Toc385950561)

[索引的定义 1](#_Toc385950562)

[创建索引 的语法 1](#_Toc385950563)

[主键和索引的关系 2](#_Toc385950564)

[索引创建的原则 2](#_Toc385950565)

[索引对Select 和DML的影响 2](#_Toc385950566)

[不适合建立索引的情况 2](#_Toc385950567)

[索引失效的情况 3](#_Toc385950568)

[在线创建索引 3](#_Toc385950569)

[分区表的意义 4](#_Toc385950570)

[程序的调优 4](#_Toc385950571)

[注临时表空间的作用 5](#_Toc385950572)

[其它 5](#_Toc385950573)

[快速拷贝一张表结构 5](#_Toc385950574)

## 索引的定义

一种数据库对象（index）它一个独立存在的物理结构。.数据库 用它来达到快速根据索引字段的查询的目的。索引对使用者来说是透明的（自动的起作用）。我们在经常作为查询字段的列上建立索引。

索引和表中数据的关系就是书目录和书内容的关系。（在大数据量的表上创建索引）

索引的意义在于提高根据索引字段查询效率

## 创建索引 的语法

*--创建一个索引*

create index index\_test\_id on db\_index\_test(index\_id);

*--删除一个索引*

drop index index\_test\_id;

## 主键和索引的关系

问题：给表创建一个主键可以提高根据主键查询的效率吗？

答：会提高根据主键查询的效率。这是由于当表中字段添加唯一约束时，数据库会为这个字段自动添加一个索引。而添加主键约束相当于添加了not null+ unique 约束，那么数据库自然会在主键列自动创建索引。所以提高了根据主键列查询的效率。但是提高效率的并不是主键约束而是自动创建的那个索引。

主键：用来唯一标识表中一条记录。它本身和提高查询效率是没有关系的。

创建一个主键就相当于在该列上声明一个非空约束和唯一约束。

但是在Oracle中当创建一个唯一约束时数据库将自动为该列创建一个唯一索引

其实提高查询效率就是这个索引起到的作用。

### 索引创建的原则

在经常作为查询条件的字段上创建索引。因为创建索引提升了对这个表的查询效率但是大大降低了DML语句的效率;

### 索引对Select 和DML的影响

索引提高了根据索引字段查询的效率。但是在一个表中过多的建立索引会降低DML的执行效率，因为在改变数据的同时还将要维护索引数据。

## 不适合建立索引的情况

1. 表中数据比较少（至少10万条）

2、大量重复数据的字段上适合建立位图索引

create bitmap index INDEXSDFSD on DB\_INDEX\_TEST (INDEX\_CITY)

3、频繁进行DML操作的表不适合建立过多的索引。(如：话费查询)

## 索引失效的情况

1. 没有按建立索引的列进行条件查询
2. 在查询列上使用了函数。如果在查询时经常在列上进行函数运算则需要针对这个函数建立索引

select \* from db\_index\_test t where

substr(db\_name,1,10)='910EC29B40'

如果是依托于转换函数建立的索引，在查询时使用的格式声明一定要和建立索引中的声明完全一致（包括大小写）。

1. 模糊查询%前置失效
2. 当查询条件为is null时失效。当列中有数的情况下有的字段为null该字段不会被放入索引树 中
3. 当使用to\_char,to\_date 当格式大小写和索引定义不一致时，索引失效

select \* from db\_index\_test where

to\_char(db\_rq,'YYYY-MM-DD') ='1978-05-02'

6、对索引列进行运算导致索引失效,我所指的对索引列进行运算包括(+，-，\*，/， 等)

错误的例子：select \* from test where id-1=9;

正确的例子：select \* from test where id=10;

7、我们可以为经常作为组合查询的条件建立一个联合索引

单独引用复合（联合索引）索引里非第一位置的索引列

create index index\_dssds on db\_test\_index(index\_no,index\_name)

对index\_name进行查询时失效.

用于联合索引的列不能使用 or 关联只能用and

select \* from db\_test\_index where index\_no=’sdfsdfsdfsd’ or index\_name=’sdfsdfsdf’

是失效的

## 创建索引的时机

因为默认创建索引时，不允许对表进行DML语句操作。当系统使用效率最低时。创建索引能使损失降到最低。或者创建在线索引（耗时会更多）。

## 在线创建索引

create index index\_id\_name on

db\_index\_test(db\_id,db\_name) online

--重构索引

alter index index\_db\_index\_test rebuild

支持同时对数据表的操作，否则创建索引的时候表会被锁定

## 分区表的意义

将表中信息区域化

分区表和索引和表中数据的关系：就像书的卷数和每卷的目录和具体内容之间的关系一样。

注意:只有在数据量特别大（千万条以上）的时候才考虑创建分区表。

分区的方式：列表分区、范围分区、哈希分区

### 程序的调优

1. 在Java和数据库sql的算法上考虑：
2. 查询效率低的话：1）是否在查询条件列上创建了索引 2）索引是否失效了3）表中数据特别大是否可以考虑建分区表啊
3. DML语句的效率的话:1)索引过多会降低索引
4. 表空间：数据表空间：增大会提高查询和DML效率

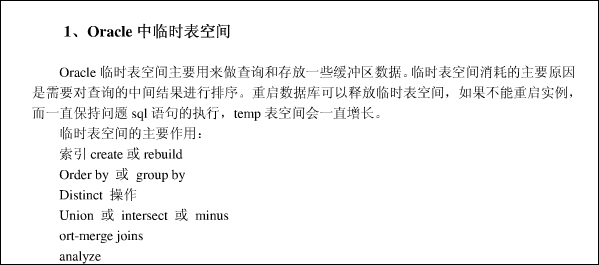
临时表空间：如果查询的数据结果比较大并需要进行排序分组等操作。或者是创建索引

则可以考虑扩大临时表空间

undo表空间：如果DML修改的数量比较多扩大UNDO表空间

1. 表空间管理的数据文件的增量上：初始化就大点，增量大点；
2. 硬件更换

### 注临时表空间的作用



## 其它

### 快速拷贝一张表结构

create table temp\_dd as

select \* from db\_test\_index where 1!=1

*--快速建表并插入*

**create** **table** **dept\_dy**

**as**

**select** \* **from** **dept**

*--快速清空一张表*

**truncate** **table** **dept\_dy**

*--快速插入*

**insert** **into** **dept\_dy**(**deptno**,**dname**,**loc**)

**select** **deptno**,**dname**,**loc** **from** **dept**

*--查询并修改*

**select** \* **from** **dept** **for** **update**

**select** **t.**\*,**rowid** **from** **dept** **t**

***第九部分 Oracle经典面试题***

这也许是你一直期待的文章，在关注这部分技术问题的同时，请务必阅读有关面试中有关个人的问题和解答。这里的回答并不是十分全面，这些问题可以通过多个角度来进行解释，也许你不必在面试过程中给出完全详尽的答案，只需要通过你的解答使面试考官了解你对ORACLE概念的熟悉程度。

　　1.解释冷备份和热备份的不同点以及各自的优点

　　解答:热备份针对归档模式的数据库，在数据库仍旧处于工作状态时进行备份。而冷

　　备份指在数据库关闭后，进行备份，适用于所有模式的数据库。热备份的优点在于当备份

　　时，数据库仍旧可以被使用并且可以将数据库恢复到任意一个时间点。冷备份的优点在于

　　它的备份和恢复操作相当简单，并且由于冷备份的数据库可以工作在非归档模式下,数据库

　　性能会比归档模式稍好。(因为不必将archive log写入硬盘)

　　2.你必须利用备份恢复数据库，但是你没有控制文件，该如何解决问题呢?

　　解答:重建控制文件，用带backup control file 子句的recover 命令恢复数据库。

　　3.如何转换init.ora到spfile?

　　解答:使用create spfile from pfile 命令.

　　4.解释data block , extent 和 segment的区别(这里建议用英文术语)

　　解答:data block是数据库中最小的逻辑存储单元。当数据库的对象需要更多的物理

　　存储空间时，连续的data block就组成了extent . 一个数据库对象拥有的所有extents被

　　称为该对象的segment.

　　5.给出两个检查表结构的方法

　　解答:1.DESCRIBE命令

　　2.DBMS\_METADATA.GET\_DDL 包

　　6.怎样查看数据库引擎的报错

　　解答:alert log.

　　7.比较truncate和delete 命令

　　解答:两者都可以用来删除表中所有的记录。区别在于:truncate是DDL操作，它移动

　　HWK，不需要rollback segment .而Delete是DML操作, 需要rollback segment 且花费较长

　　时间.

　　8.使用索引的理由

　　解答:快速访问表中的data block

　　9.给出在STAR SCHEMA中的两种表及它们分别含有的数据

　　解答:Fact tables 和dimension tables. fact table包含大量的主要的信息而dime

　　nsion tables 存放对fact table 某些属性描述的信息

　　10.FACT Table上需要建立何种索引?

　　解答:位图索引 (bitmap index)

　　11. 给出两种相关约束?

　　解答:主键和外键

　　12. 如何在不影响子表的前提下，重建一个母表

　　解答:子表的外键强制实效，重建母表，激活外键

　　13. 解释归档和非归档模式之间的不同和它们各自的优缺点

　　解答:归档模式是指你可以备份所有的数据库 transactions并恢复到任意一个时间点

　　。非归档模式则相反，不能恢复到任意一个时间点。但是非归档模式可以带来数据库性能

　　上的少许提高.

　　14. 如何建立一个备份控制文件?

　　解答:Alter database backup control file to trace.

　　15. 给出数据库正常启动所经历的几种状态 ?

　　解答:STARTUP NOMOUNT ?C 数据库实例启动

　　STARTUP MOUNT - 数据库装载

　　STARTUP OPEN ?C 数据库打开

　　16. 哪个column可以用来区别V$视图和GV$视图?

　　解答:INST\_ID 指明集群环境中具体的 某个instance 。

　　17. 如何生成explain plan?

　　解答:运行utlxplan.sql. 建立plan 表

　　针对特定SQL语句，使用 explain plan set statement\_id = 'tst1' into pl

　　an\_table

　　运行utlxplp.sql 或 utlxpls.sql察看explain plan

　　18. 如何增加buffer cache的命中率?

　　解答:在数据库较繁忙时，适用buffer cache advisory 工具，查询v$db\_cache\_adv

　　ice.如果有必要更改，可以使用 alter system set db\_cache\_size 命令

　　19. ORA-01555的应对方法?

　　解答:具体的出错信息是snapshot too old within rollback seg , 通常可以通过增

　　大rollback seg来解决问题。当然也需要察看一下具体造成错误的SQL文本

　　20. 解释$ORACLE\_HOME和$ORACLE\_BASE的区别?

　　解答:ORACLE\_BASE是oracle的根目录，ORACLE\_HOME是oracle产品的目录。