目录

[一、Oracle体系结构的概览... 2](#t0)

[二、Oracle高级查询... 10](#t1)

[三、Oracle触发器... 18](#t2)

[四、Oracle存储过程... 26](#t3)

[五、Oracle事务... 30](#t4)

[六、Oracle锁... 37](#t5)

[七、Oracle包... 40](#t6)

[八、Oracle游标... 42](#t7)

[九、Oracle函数... 47](#t8)

[十、Oracle备份与恢复... 129](#t9)

[十一、Oracle故障与恢复... 134](#t10)

[十二、Oracle运维... 145](#t11)

**一、OracleOracle体系结构的概览**

更详细的一张结构图如下

Oracle服务器

由Oracle实例和Oracle数据库两大部分组成。

Oracle实例

是一种数据库访问机制，主要由内存结构和进程结构组成。

内存结构主要包括系统全局区（System Global Area，SGA）、进程全局区（Process Global Area，PGA）等

实例的后台进程有5个是必须的

SMON 系统监视器进程

PMON 进程监视器进程

DBWR 数据库书写器

LGWR 日志书写器

CKPT 检查点进程

每个实例只能操作其对应的一个数据库，但一个数据库可以同时被几个实例操作（RAC）

Oracle数据库

由以下三种操作系统文件组成：

控制文件（control files）

数据文件（data files）

重做日志文件（redo log files）

其它服务器文件：

初始化参数文件（parameter files）

口令文件（password files）

归档重做日志文件（archived redo log files）

与Oracle服务器的连接

一个连接即称为一个会话，连接可分为以下两种：

1、专用服务器连接

一个用户进程对应创建一个服务器进程，用户进程与服务器进程是一对一的关系。

2、共享服务器连接

多个用户进程同时对应一个服务器进程。

各种不同的连接方式

1、基于主机方式

用户进程与服务器进程运行在同一台计算机相同的操作系统下，用户进程与Oracle服务器的通信是通过操作系统内部的进程通信（inter processcommunication，IPC）机制来建立的。

2、客户端-服务器（client-server）（两层模型）方式

用户进程与Oracle服务器的通信是通过网络协议（如TCP/IP）来完成的。

3、客户端-应用服务器-服务器（client-application server-server）（三层模型）方式

用户的个人计算机通过网络与应用服务器或网络服务器进行通信，该应用服务器或网络服务器同样再通过网络与运行数据库的计算机连接。

Oracle执行SQL查询语句的步骤

1、SQL正文放入共享池（shared pool）的库缓存（library cache）。

2、检查是否有相同的SQL正文，没有就进行以下编译处理，否则跳过。

1）语法检查

2）通过数据字典检查表和列的定义

3）对所操作的对象加编译锁，防止编译期间的对象定义被改变

4）检查用户权限

5）生成执行计划

6）将编译后的代码和执行计划放入共享SQL区

3、执行

由服务器进程执行SQL语句。

4、提取数据

由服务器进程选择所需的数据行，需要时排序（PGA中），返回给用户进程。

系统全局区（System Global Area，SGA）

SGA包含以下几大块：

固定区域（Fixed Size）：存储SGA中各个组件的信息，大小不能修改

可变区域（Variable Size）：包括共享池、大池、流池、JAVA池

数据库高速缓冲区缓存（Database buffer cache）：大小由参数db\_cache\_size指定（10g后参数db\_cache\_size默认为0）

重做日志缓冲区缓存（Redo log buffer cache）：大小通常大于参数log\_buffer的设置，因为在内存中还要设置保护页对log buffer进行保护。

以下命令可以看到SGA的内存分配概览

show sga

Total System Global Area 4960579584 bytes

FixedSize                 2184232 bytes

VariableSize           2902461400 bytes

Database Buffers        2046820352 bytes

RedoBuffers               9113600 bytes

或者

select \* from v$sga;

NAME                     VALUE

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

FixedSize             2184232

Variable Size        2902461400

Database Buffers     2046820352

RedoBuffers           9113600

共享池（shared pool）

用于存放SQL语句、PL/SQL代码、数据字典、资源锁和其他控制信息。它由初始化参数SHARED\_POOL\_SIZE控制其大小。它包含以下几个缓冲区：

1、数据字典缓存（datadictionary cache）：用于存储经常使用的数据字典信息。比如（表的定义、用户名、口令、权限、数据库的结构等）。Oracle运行过程中经常访问该缓存以便解析SQL语句，确定操作的对象是否存在，是否具有权限等。如果不在数据字典缓存中，服务器进程就从保存数据字典信息的数据文件中将其读入到数据字典缓存中。数据字典缓存中保存的是一条一条的记录（就像是内存中的数据库），而其他缓存区中保存的是数据块信息。

2、库缓存（LibraryCache）：用于保存最近解析过的SQL语句、PL/SQL过程。Oracle在执行一条SQL语句、一段PL/SQL过程前首先在库缓存中搜索，如果查到它们已经解析过了，就利用库缓存中的解析结果和执行计划来执行，而不必重新对它们进行解析，显著提高执行速度。Oracle是通过比较SQL或PL/SQL语句的正文来决定两个语句是否相同的，只有正文完全相同，Oracle才重用已存在的编译后的代码和执行计划。应该尽量用绑定变量的方式写SQL，绑定变量不是在编译阶段赋值的，而是在运行阶段赋值的，因此语句可以不用重新编译。

库缓存的管理采用LRU（least recently used）的队列算法，即最近最少使用的队列算法。刚使用的内存块放在LRU队列的头部，而进程每次从队列的尾部获取内存块，获取到的内存块立即移至队列头部。最终使长时间没有使用到的内存块自然移到了队列的尾部而被最先使用。

Oracle没有提供单独设置库缓存或数据字典缓存空间大小的方法，而是通过设置共享池的大小来间接设置，通过参数SHARED\_POOL\_SIZE可调整，其大小受限于SGA的尺寸SGA\_MAX\_SIZE参数。

ORACLE将每一条SQL语句分解为可共享、不可共享的两部分。

共享SQL区：存储的是最近执行的SQL语句、解析后的语法树和优化后的执行计划。这样以后执行相同的SQL语句就直接利用在共享SQL区中的缓存信息，不必重复语法解析了。Oracle在执行一条新的SQL语句时，会为它在共享SQL区中分配空间，分配的大小取决于SQL语句的复杂度。如果共享SQL区中没有空闲空间，就利用LRU算法，释放被占用的空间。

私有SQL区（共享服务器模式）：存储的是在执行SQL语句时与每个会话或用户相关的私有信息。其他会话即使执行相同的SQL语句也不会使用这些信息。比如绑定变量、环境和会话参数。

3、SQL和PL/SQL结果缓存：此高速缓存用于存储SQL查询或PL/SQL函数的结果，以加快其将来的执行速度。

4、锁与其他控制结构：存储ORACLE例程内部操作所需的信息，如各种锁、闩、寄存器值等。

数据库高速缓冲区缓存（Database Buffer Cache）

也叫块缓存区，用于存放从数据文件读取的数据块，其大小由初始化参数DB\_CACHE\_SIZE决定。

工作原理是通过LRU队列（最近最少使用Least Recently Used）。查询时，Oracle会先把从磁盘读取的数据放入内存供所有用户共享，以后再查询相关数据时不用再次读取磁盘。插入和更新时，Oracle会先在该区域中缓存数据，之后批量写到硬盘中。通过块缓冲区，Oracle可以通过内存缓存提高磁盘的I/O性能。

数据高速缓存块由许多大小相等的缓存块组成，这些缓存块分为3大类：

1）脏缓存块（Dirtybuffers）：脏缓存块中保存的是被修改过的缓存块。即当一条SQL语句对某个缓存块中的数据进行修改后，该缓存块就被标记为脏缓存块。最后该脏缓存块被DBWn进程写入到硬盘的数据文件中永久保存。

2）命中缓存块（Pinnedbuffers）：命中缓存块中保存的是最近正在被访问的缓存块。它始终被保留在数据高速缓存中，不会被写入数据文件。

3）空闲缓存块（Freebuffers）：该缓存块中没有数据，等待被写入数据。oracle从数据文件中读取数据后，寻找空闲缓存块，以便写入其中。

Oracle通过2个列表（DIRTY、LRU）来管理缓存块：

1）DIRTY列表中保存已经被修改但还没有被写入到数据文件中的脏缓存块。

2）LRU列表中保存所有的缓存块（还没有被移动到DIRTY列表中的脏缓存块、空闲缓存块、命中缓存块）。当某个缓存块被访问后，该缓存块就被移动到LRU列表的头部，其他缓存块就向LRU列表的尾部移动。放在最尾部的缓存块就最先被移出LRU列表。

数据高速缓存的工作原理过程：

1）ORACLE在将数据文件中的数据块复制到数据高速缓存之前，先在数据高速缓存中找空闲缓存块，以便容纳该数据块。Oracle 将从LRU列表的尾部开始搜索，直到找到所需的空闲缓存块为止。

2）如果先搜索到的是脏缓存块，将该脏缓存块移动到DIRTY列表中，然后继续搜索。如果搜索到的是空闲缓存块，则将数据块写入，然后将该缓存块移动到DIRTY列表的头部。

3）如果能够搜索到足够的空闲缓存块，就将所有的数据块写入到对应的空闲缓存块中，搜索写入过程结束。

4）如果没有搜索到足够的空闲缓存块，则ORACLE就先停止搜索，而是激活DBWn进程，开始将DIRTY列表中的脏缓存块写入到数据文件中。

5）已经被写入到数据文件中的脏缓存块将变成空闲缓存块，并被放入到LRU列表中。执行完成这个工作后，再重新开始搜索，直到找到足够的空闲缓存块为止。

这里可以看出，如果你的高速缓冲区很小的，不停地写写，造成很大I/O开销。

块缓冲区可以配置1、2或3个缓冲池，默认只有第一个：

1）默认池（Defaultpool）：所有数据默认都在这里缓存，除非你在建表的时候指定 Store(buffer\_pool keep)or Store(buffer\_pool recycle)。使用LRU算法管理。

2）保持池（Keep pool）：缓存需要多次重用的数据，长期保存在内存中，缺省值为0。

3）回收池（Recyclepool）：用来缓存很少重用的数据，用完就释放，缺省值为0。

原来只有一个默认池，所有数据都在这里缓存。这样会产生一个问题：大量很少重用的数据会把需重用的数据“挤出”缓冲区，造成磁盘I/O增加，运行速度下降。后来分出了保持池和回收池根据是否经常重用来分别缓存数据。这三部分内存池需要手动确定大小，并且之间没有共享。例如：保持池中已经满了，而回收池中还有大量空闲内存，这时回收池的内存不会分配给保持池，这些池一般被视为一种非常精细的低级调优设备，只有所有其他调优手段大多用过之后才应考虑使用。

在9i之前，数据缓冲区的大小是由DB\_BLOCK\_BUFFER决定的，缓冲区的大小为DB\_BLOCK\_SIZE（Oracle数据块大小，创建数据库时设定好后续不能改变）和DB\_BLOCK\_BUFFERS（缓冲区块的个数）这两个参数的乘积，改变需重启数据库。之后的版本则是由参数DB\_CACHE\_SIZE及DB\_nK\_CACHE\_SIZE确定。不同的表空间可以使用不同的块大小，在创建表空间中加入参数BLOCKSIZE指定该表空间数据块的大小，如果指定的是2k，则对应的缓冲区大小为DB\_2K\_CACHE\_SIZE参数的值，如果指定的是4k，则对应的缓冲区大小为DB\_4K\_CACHE\_SIZE参数的值，以此类推。如果不指定BLOCKSIZE，则默认为参数DB\_BLOCK\_SIZE的值，对应的缓冲区大小是DB\_CACHE\_SIZE的值。

数据库高速缓冲区缓存大小的建议可以参看内存缓冲区顾问v$db\_cache\_advice

打开或关闭该顾问

alter system set db\_cache\_advice=on|off|ready;

重做日志缓冲区（Redo Log Buffer Cache）

Oracle在DML或DDL操作改变数据写到数据库高速缓冲区缓存之前，先写入重做日志缓冲区，随后LGWR后台进程再把日志条目写到磁盘上的联机日志文件中。日志缓冲区的大小由初始化参数log\_buffer决定大小。

以下情况触发LGWR进程将日志缓存数据写到联机日志文件中：

1）每隔3秒

2）缓存达到1MB或1/3满时

3）用户提交时

4）缓冲区的数据写入磁盘前

大池（Large Pool）

可以根据实际业务需要来决定是否在SGA区中创建大池。如果没有创建大池，则需要大量内存空间的操作将占用共享池的内存, 将对系统性能带来影响。

大池没有LRU队列，在共享服务器连接时，PGA的大部分区域（UGA）将放入大池（不包括堆栈区域），并行化的数据库操作、大规模的I/O及备份和恢复操作可能用到大池。大池由初始化参数LARGE\_POOL\_SIZE确定其大小。

流池（Stream Pool）

加强对流的支持，大小由参数STREAM\_POOL\_SIZE确定。流池（如果没有配置流池，则是共享池中至多10%的空间）用于缓存流进程在数据库间移动/复制数据时使用的队列消息。

Java池（Java Pool）

用于支持在数据库中运行java代码，大小由参数JAVA\_POOL\_SIZE确定。

进程全局区(Porcess Global Area, PGA)

一个PGA是一块独占内存区域，Oracle进程以专有的方式用它来存放数据和控制信息。当Oracle进程启动时，PGA也就由Oracle数据库创建了。当用户进程连接到数据库并创建一个对应的会话时，Oracle服务进程会为这个用户专门设置一个PGA区，用来存储这个用户会话的相关内容。当这个用户会话终止时，系统会自动释放这个PGA区所占用的内存。这个PGA区对于数据库的性能有比较大的影响，特别是对于排序操作的性能。

PGA主要包含排序区、会话区、堆栈区和游标区四个部分。通常情况下，系统管理员主要关注的是排序区，在必要时需要手工调整这个排序区的大小。游标区是一个动态的区域，在游标打开时创建，关闭时释放，故在数据库开发时，不要频繁的打开和关闭游标可以改善数据库的性能。其他分区的内容管理员只需要了解其用途，日常的维护交给数据库系统来完成即可。

1、为排序设置合理的排序区大小

当用户需要对数据进行排序时，系统会将需要排序的数据保存到PGA中的一个排序区内，然后在这个排序区内对这些数据进行排序。如需要排序的数据有2M，那么排序区内必须至少要有2M的空间来容纳这些数据。然后排序过程中又需要有2M的空间来保存排序后的结果。由于系统从内存中读取数据比从硬盘中读取数据的速度要快几千倍，为此如果这个数据排序与读取的操作都能够在内存中完成，无疑可以在很大程度上提高数据库排序与访问的性能。如果这个排序的操作都能够在内存中完成，显然这是很理想的。但是如果PGA中的排序区容量不够，不能容纳排序后的数据，系统会从硬盘中获取一个空间，用来保存需要排序的数据。此时排序的效率就会降低许多。为此在数据库管理中，如果发现用户的很多操作都需要用到排序，那么为用户设置比较大的排序区，可以提高用户访问数据的效率。

在Oracle数据库中，这个排序区主要用来存放排序操作产生的临时数据。一般来说，这个排序区的大小占据PGA程序缓存区的大部分空间，这是影响PGA区大小的主要因素。在小型应用中，数据库管理员可以直接采用其默认的值。但是在一些大型的应用中，或者需要进行大量记录排序操作的数据库系统中，管理员可能需要手工调整这个排序区的大小，以提高排序的性能，这可以通过初始化参数SORT\_AREA\_SIZE来实现。

2、会话区保存着用户的权限等重要信息

会话区保存了会话所具有的权限、角色、性能统计等信息，通常都是由数据库系统自我维护，管理员不用干预。当用户进程与数据库建立会话时，系统会将这个用户的相关权限查询出来，保存在这个会话区内。用户进程在访问数据时，系统会核对会话区内的用户权限信息，看看其是否具有相关的访问权限。

3、堆栈区保存变量信息

保存着绑定变量、会话变量、SQL语句运行时的内存结构等重要的信息。通常都是由数据库系统自我维护，管理员不用干预。这些分区的大小，也是系统根据实际情况来进行自动分配的。当这个用户会话结束时，系统会自动释放这些区所占用的空间。

4、游标区

游标区是一个动态的区域。当用户执行游标语句打开游标时，系统会在PGA中创建游标区，当关闭游标时，这个区域就会被释放。创建与释放需要占用一定的系统资源，花费一定的时间，如果频繁的打开和关闭游标，就会降低语句的执行性能。所以在写语句时，游标最好不要频繁的打开和关闭。

初始化参数OPEN\_CURSORS可以根据实际需要来设置，控制用户能够同时打开游标的数量。在确实需要游标的情况下，如果硬件资源支持的话，也可以放宽这个限制。

用户全局区（User Global Area, UGA）

专用服务器模式下，进程和会话是一对一的关系，UGA被包含在PGA中，在共享服务器模式下，进程和会话是一对多的关系，所以UGA就不再属于PGA了，而会在大型池（LargePool）中分配。但如果从大型池中分配失败，如大型池太小，或是根本没有设置大型池，则从共享池（SharedPool）中分配。

实例的后台进程

1、重做日志写进程LGWR

Oracle使用快速提交的技术，保证系统的效率，并保证系统崩溃时所提交的数据可以得到恢复，引入系统改变号SCN。

SCN是单调递增的正整数，与Oracle内部时间戳对应，保证系统中数据的同步和读一致性。

发出commit命令后：

1）服务器进程把提交的记录连同产生的SCN号一起写入重做日志缓存。

2）LGWR把缓存中一直到提交的记录和SCN连续的写入联机重做日志文件中。在此之后，Oracle就能够保证即使在系统崩溃的情况下所有已提交的数据也可以得到恢复（联机重做日志文件在归档前不能被覆盖重写）。

3）Oracle通知用户进程提交已经完成。

4）服务器进程修改数据库高速缓存中的数据状态，释放资源和打开锁。

写日志要比写数据效率高，记录格式紧凑，I/O量少，顺序写入。

LGWR写入时机：

1）事务被提交

2）日志缓存中变化记录超过1MB

3）日志缓存中的记录超过缓冲区容量的1/3

4）DBWR写入数据文件之前

5）每3秒钟

日志挖掘器（ log miner）工具，可将日志文件中的内容转化为用户能够理解的正文信息。

2、数据库写进程DBWR/DBWn

Oracle实例允许启动最多10个数据库写进程DBW0～DBW9

DBWn写入时机：

1）当脏缓冲区的数量超过了所设定的限额

2）所设定的时间间隔已到

3）当有进程需要数据库高速缓冲区却找不到空闲的缓冲区时

4）当检查点发生时

5）当某个表被删除或截断时

6）当某个表空间被设置为只读时

7）当对某个表空间进行联机备份时

8）当某个临时表空间被设置为脱机状态或正常状态时

3、系统监视器进程SMON

当Oracle系统由于某种原因如断电，SGA中已经提交但还未被写入数据文件中的数据将丢失。当数据库重启时，系统监视器进程SMON将自动执行Oracle实例的恢复工作。

1）执行前滚，将已提交到重做日志文件中但还未写到数据文件中的数据写到数据文件中。（通过SCN号识别提交记录）

2）前滚完成后立即打开数据库，这时数据文件中可能还有一些没有提交的数据。（之所以这样安排，主要是为了提高系统的效率）

3）回滚未提交的事务（数据）

4）执行一些磁盘空间的维护工作

4、进程监视器进程PMON

当某个用户进程崩溃时（如未正常退出），进程监视器进程PMON将负责清理工作。

1）回滚用户当前的事务

2）释放用户所加的所有表一级和行一级的锁

3）释放用户所有的其它资源

5、检查点进程CKPT

Oracle为了提高系统效率和保证数据库的一致性，引入检查点事件。

DBWR将SGA中所有已改变了的数据库高速缓冲区缓存中的数据（包括已提交的和未提交的）写到数据文件中时，将产生检查点事件。

保证了所有到检查点为止的变化了的数据都已经写到了数据文件中，在实例恢复时检查点之前的重做日志记录已经不再需要，从而加快了实例的恢复速度。

检查点事件发生时，Oracle要将检查点号写入数据文件头中，还要将检查点号、重做日志序列号、归档日志名称和SCN号都写入控制文件中。

过于频繁的检查点会使联机操作受到冲击，因此需要在实例的恢复速度和联机操作之间折中。（大多在20分钟以上）

检查点发生时机：

1）重做日志的切换

2）LOG\_CHECKPOINT\_TIMEOUT 这个延迟参数的到达

3）相应字节(LOG\_CHECKPOINT\_INTERVAL\* size of IO OS blocks)被写到当前的重做日志

4）ALTER SYSTEM SWITCH LOGFILE命令

5）ALTER SYSTEM CHECKPOINT命令

查看数据库的检查点号：

Select checkpoint\_change# fromv$database;

查看数据库当前的SCN号：

Select current\_scn from v$database;

6、归档日志进程ARCH/ARCn

当数据库运行在归档日志模式下时，ARCH/ARCn进程将把日志切换后的联机重做日志文件中的数据复制到归档日志文件中，保证不会因致联机日志文件组的循环切换而导致日志数据丢失，从而保证数据库的可完全恢复。

归档日志文件是脱机的。

Oracle确保在一组重做日志的归档操作完成之前不会重新使用该组重做日志。

**二、Oracle高级查询**

涉及内容

　　1.掌握SELECT语句的多表连接查询。

　　2.掌握SELECT语句的子查询。

具体操作

根据Oracle数据库scott方案下的emp表和dept表，完成下列操作：

  1.查询所有工种为CLERK的员工的姓名及其部门名称。

select ename,dname   
from scott.emp t1 innerjoin scott.dept t2 ont1.deptno=t2.deptno   
where job='CLERK';

  2.查询所有部门及其员工信息，包括那些没有员工的部门。

select \* from scott.emp t1 right join scott.deptt2 on t1.deptno=t2.deptno

  3.查询所有员工及其部门信息，包括那些还不属于任何部门的员工。

select \* from scott.emp t1 left join scott.deptt2 on t1.deptno=t2.deptno

  4.查询在SALES部门工作的员工的姓名信息。

用子查询实现:

select \* from scott.emp

where deptno=(select deptno from scott.dept wheredname='SALES')

用连接查询实现:

select \* from scott.emp t1 inner join scott.deptt2 on t1.deptno=t2.deptno  
where t2.dname='SALES';

注意两种实现方式，在行和列上的变化。

  5.查询所有员工的姓名及其直接上级的姓名。

select t1.ename as 员工姓名,t2.ename 经理姓名 fromscott.emp t1,scott.emp t2

where t1.mgr=t2.empno;

  6.查询入职日期早于其上级领导的所有员工的信息。

select t1.ename as 员工姓名,t2.ename 经理姓名 fromscott.emp t1,scott.emp t2

where t1.mgr=t2.empno andt1.hiredate<t2.hiredate

  7.查询从事同一种工作但不属于同一部门的员工信息。

selectt1.ename,t1.job,t1.deptno,t2.ename,t2.job,t2.deptno fromscott.emp t1 cross join scott.emp t2

 where t1.job=t2.job and t1.deptno<>t2.deptno

  8.查询10号部门员工及其领导的信息。

select t1.ename as 员工姓名,t2.ename 经理姓名 fromscott.emp t1,scott.emp t2

where t1.mgr=t2.empno andt1.deptno=**10**;

  9.使用UNION将工资大于2500的雇员信息与工作为ANALYST的雇员信息合并。

select \* from scott.emp where sal>**2500**

union

select \* from scott.emp where job='ANALYST'

 10.通过INTERSECT集合运算，查询工资大于2500，并且工作为ANALYST的雇员信息。

select \* from scott.emp where sal>**2500**

intersect

select \* from scott.emp where job='ANALYST

 11.使用MINUS集合查询工资大于2500，但工作不是ANALYST的雇员信息。

select \* from scott.emp where sal>**2500**

minus

select \* from scott.emp where job='ANALYST';

 12.查询工资高于公司平均工资的所有员工信息。

select \* from scott.emp where sal>(select avg(sal) from scott.emp)

 13.查询与SMITH员工从事相同工作的所有员工信息。

select \* from scott.emp where job=(select job from scott.emp

where ename='SMITH')

 14.查询工资比SMITH员工工资高的所有员工信息。

select \* from scott.emp where sal>(select  sal from scott.emp

where ename='SMITH')

 15.查询比所有在30号部门中工作的员工的工资都高的员工姓名和工资。

select ename,sal from scott.emp wheresal>all(select  sal fromscott.emp where deptno=**30**)

 16.查询部门人数大于5的部门的员工信息。

select \* from scott.emp   
where deptno in(select deptno fromscott.emp group bydeptno having count(\*)>**5**);

 17.查询所有员工工资都大于2000的部门的信息。

select \* from scott.dept

where deptno in(select deptno from scott.emp group by deptno having min(sal)>**2000**)

 18.查询人数最多的部门信息。

select \* from scott.dept

where deptno in (select deptno from (select deptno,count(\*)

                         as人数

                         fromscott.emp

                         group by deptno)

where 人数=(selectmax(人数)

                 from(select deptno,count(\*)

                 as人数

                 fromscott.emp

                 group by deptno)));

 19.查询至少有一个员工的部门信息。

select \* from scott.dept

where deptno in(select deptno

 　　　　　　　　　　from scott.emp

group by deptno

having count(\*)>=**1**)

 20.查询工资高于本部门平均工资的员工信息。

select \* from scott.emp e

where sal>(select avg(sal)

 fromscott.emp

　　　　　　group by deptno

having e.deptno=deptno);

 21.查询工资高于本部门平均工资的员工信息及其部门的平均工资。

select \* from((select \* from scott.emp e

                     where sal>(select avg(sal)

                     fromscott.emp

                     group by deptno

                     having e.deptno=deptno)) t1

                     inner join

                   (select avg(sal),deptno

                    fromscott.emp

                    group by deptno) t2 ont1.deptno=t2.deptno);

 22.查询每个员工的领导所在部门的信息。

select \* from scott.dept

where deptno in(select distinct deptno

fromscott.emp

where empno in(select distinct mgr

from scott.emp));

 23.查询平均工资低于2000的部门及其员工信息。

select \* from scott.emp t1,scott.dept t2

where t1.deptno=t2.deptno andt1.deptno in(select deptno

        fromscott.emp

group by deptno

having avg(sal)<**2000**)

习题

  1.如果需要将雇员表中的所有行连接到雇员表中的所有行，则应创建哪种类型的连接？**（B）**

   A.等值连接    B.笛卡尔乘积 C.内连接  D.外连接

  2.如果需要从顾客表和订单表中查询所有顾客及其下达的所有订单，并且要求查询结果中先按顾客所在公司名称的升序排列，再按订单金额的降序排列。应执行以下哪条语句？**（B）**

A.SELECT c.顾客标识,c.公司名称,o.订单日期,o.顾客标识,o.金额

  FROM 顾客c,订单 o

  WHERE c.顾客标识=o.顾客标识

  ORDER BY 金额DESC, 公司名称;

B. SELECT c.顾客标识,c.公司名称,o.订单日期,o.顾客标识,o.金额

  FROM 顾客c,订单 o

  WHERE c.顾客标识=o.顾客标识

  ORDER BY公司名称,金额 DESC;

C. SELECT c.顾客标识,c.公司名称,o.订单日期,o.顾客标识,o.金额

  FROM 顾客c,订单 o

  WHERE c.顾客标识=o.顾客标识

  ORDER BY公司名称,金额;

D. SELECT c.顾客标识,c.公司名称,o.订单日期,o.顾客标识,o.金额

  FROM 顾客c,订单 o

  WHERE c.顾客标识=o.顾客标识

  ORDER BY公司名称 ASC,金额 DESC;

 3.评估以下SQL语句：

   SELECT e.雇员标识,e.姓氏,e.名字,d.部门名称

  FROM 雇员e,部门 d

  WHERE e.部门标识=d.部门标识

  AND 雇员.部门标识>5000

  ORDER BY 4;

哪个字句的语法有错误？**（E）**

A. SELECT e.雇员标识,e.姓氏,e.名字,d.部门名称

B. FROM 雇员e,部门 d

C. WHERE e.部门标识=d.部门标识

D. AND 雇员.部门标识>5000

E. ORDER BY 4;

 4.评估以下语句：

   SELECT 部门标识,AVG(薪金)

  FROM 雇员

  WHERE 职务标识<> 69 879

  GROUP BY 部门标识

  HAVING AVG(薪金)>35000

  ORDER BY部门标识;

哪些子句限制了返回结果？请选择两个正确答案。**（BD）**

A. SELECT 部门标识,AVG(薪金)

B. WHERE 职务标识<> 69 879

C. GROUP BY 部门标识

D. HAVING AVG(薪金)>35000

5.在SELECT语句中各个子句的正确顺序是什么？**（C）**

A. SELECT

   FROM

   WHERE

   ORDERBY

   GROUPBY

   HAVING

B.  SELECT

   FROM

HAVING

   ORDER BY

WHERE

   GROUP BY

C.  SELECT

   FROM

WHERE

GROUP BY

HAVING

   ORDER BY

D.  SELECT

   FROM

WHERE

HAVING

   ORDER BY

   GROUP BY

 6.以下哪个运算符可用于多行子查询？**（A）**

   A.IN            B.<>           C.=            D.LIKE

 7.假设数据库中有顾客表和订单历史记录表。其中，顾客表中包括：客户标识NUMBER（5）、姓名VARCHAR2（25）、信贷限额NUMBER（8,2）、开户日期（DATE）；订单历史记录表中包括：订单标识NUMBER（5）、客户标识NUMBER（5）、订单日期（DATE）、总计NUMBER（8,2）。以下哪种方案需要使用子查询来返回需要的结果？**（D）**

A.需要显示每个顾客账户下的开户日期

B.需要显示顾客下达订单的各个日期

C.需要显示在特定日期下达的所有订单

D.需要显示与编号为25950的订单的下达日期相同的所有订单

8.如果希望在报表中显示成本值高于所有产品平均成本的产品名称，应使用以下哪些SELECT语句？**（B）A**

A. SELECT 产品名称 FROM 产品 WHERE 成本>（SELECT AVG（成本） FROM 产品）；

B. SELECT 产品名称 FROM 产品 WHERE 成本>AVG（成本）；

C. SELECT AVG（成本）, 产品名称 FROM 产品 WHERE成本> AVG（成本）GROUP BY 产品名称；

D. SELECT 产品名称 FROM（SELECT AVG（成本）FROM 产品） WHERE 成本> AVG（成本）；

9.如果单行子查询返回了空值且使用了等于比较运算符，外部查询会返回什么结果？**（B）C**

 A.不返回任何行            B.返回表中的所有行

 C.返回空值               D.返回错误

10.如果需要创建包含多行子查询的SELECT语句，可以使用哪个（些）比较运算符？**（A）**

A.IN、ANY和ALL           B.LIKE

C.BETWEEN…AND…          D.=、< 和 >

select ename,dnamefrom scott.emp t1 inner join scott.dept t2 on t1.deptno=t2.deptno wherejob='CLERK';

**三、Oracle触发器**

**触发器简介**

触发器的定义就是说某个条件成立的时候，触发器里面所定义的语句就会被自动的执行。

因此触发器不需要人为的去调用，也不能调用。

然后，触发器的触发条件其实在你定义的时候就已经设定好了。

这里面需要说明一下，触发器可以分为语句级触发器和行级触发器。

详细的介绍可以参考网上的资料，简单的说就是语句级的触发器可以在某些语句执行前或执行后被触发。而行级触发器则是在定义的了触发的表中的行数据改变时就会被触发一次。

具体举例：

1、 在一个表中定义的语句级的触发器，当这个表被删除时，程序就会自动执行触发器里面定义的操作过程。这个就是删除表的操作就是触发器执行的条件了。  
2、 在一个表中定义了行级的触发器，那当这个表中一行数据发生变化的时候，比如删除了一行记录，那触发器也会被自动执行了。

**触发器语法**

触发器的语法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | create [or replace] tigger 触发器名 触发时间 触发事件  on 表名  [for each row]  begin   pl/sql语句  end |

其中：

触发器名：触发器对象的名称。由于触发器是数据库自动执行的，因此该名称只是一个名称，没有实质的用途。  
触发时间：指明触发器何时执行，该值可取：  
before：表示在数据库动作之前触发器执行;  
after：表示在数据库动作之后触发器执行。  
触发事件：指明哪些数据库动作会触发此触发器：  
insert：数据库插入会触发此触发器;  
update：数据库修改会触发此触发器;  
delete：数据库删除会触发此触发器。  
表 名：数据库触发器所在的表。  
for each row：对表的每一行触发器执行一次。如果没有这一选项，则只对整个表执行一次。

触发器能实现如下功能：

**功能**：

1、 允许/限制对表的修改  
2、 自动生成派生列，比如自增字段  
3、 强制数据一致性  
4、 提供审计和日志记录  
5、 防止无效的事务处理  
6、 启用复杂的业务逻辑

举例

1)、下面的触发器在更新表tb\_emp之前触发，目的是不允许在周末修改表：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | create or replace trigger auth\_secure before insert or update or DELETE  on tb\_emp  begin    IF(to\_char(sysdate,'DY')='星期日') THEN      RAISE\_APPLICATION\_ERROR(-20600,'不能在周末修改表tb\_emp');    END IF;  END;  / |

2)、使用触发器实现序号自增

创建一个测试表：

|  |  |
| --- | --- |
| 1  2  3  4  5 | create table tab\_user(    id number(11) primary key,    username varchar(50),    password varchar(50)  ); |

创建一个序列：

代码如下:

create sequence my\_seq increment by 1 start with 1nomaxvalue nocycle cache 20;

创建一个触发器：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE OR REPLACE TRIGGER MY\_TGR   BEFORE INSERT ON TAB\_USER   FOR EACH ROW--对表的每一行触发器执行一次  DECLARE   NEXT\_ID NUMBER;  BEGIN   SELECT MY\_SEQ.NEXTVAL INTO NEXT\_ID FROM DUAL;   :NEW.ID := NEXT\_ID; --:NEW表示新插入的那条记录  END; |

向表插入数据:

|  |  |
| --- | --- |
| 1  2  3  4 | insert into tab\_user(username,password) values('admin','admin');  insert into tab\_user(username,password) values('fgz','fgz');  insert into tab\_user(username,password) values('test','test');  COMMIT; |

查询表结果：SELECT \* FROM TAB\_USER;

3)、当用户对test表执行DML语句时，将相关信息记录到日志表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | --创建测试表  CREATE TABLE test(    t\_id  NUMBER(4),    t\_name VARCHAR2(20),    t\_age NUMBER(2),    t\_sex CHAR  );  --创建记录测试表  CREATE TABLE test\_log(    l\_user  VARCHAR2(15),    l\_type  VARCHAR2(15),    l\_date  VARCHAR2(30)  ); |

创建触发器：

|  |  |
| --- | --- |
| 8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | --创建触发器  CREATE OR REPLACE TRIGGER TEST\_TRIGGER   AFTER DELETE OR INSERT OR UPDATE ON TEST  DECLARE   V\_TYPE TEST\_LOG.L\_TYPE%TYPE;  BEGIN   IF INSERTING THEN    --INSERT触发    V\_TYPE := 'INSERT';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功插入，并已记录到日志');   ELSIF UPDATING THEN    --UPDATE触发    V\_TYPE := 'UPDATE';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功更新，并已记录到日志');   ELSIF DELETING THEN    --DELETE触发    V\_TYPE := 'DELETE';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功删除，并已记录到日志');   END IF;   INSERT INTO TEST\_LOG   VALUES    (USER, V\_TYPE, TO\_CHAR(SYSDATE, 'yyyy-mm-dd hh24:mi:ss')); --USER表示当前用户名  END;  /  --下面我们来分别执行DML语句  INSERT INTO test VALUES(101,'zhao',22,'M');  UPDATE test SET t\_age = 30 WHERE t\_id = 101;  DELETE test WHERE t\_id = 101;  --然后查看效果  SELECT \* FROM test;  SELECT \* FROM test\_log; |

运行结果如下：

3)、创建触发器，它将映射emp表中每个部门的总人数和总工资

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | --创建映射表  CREATE TABLE dept\_sal AS  SELECT deptno, COUNT(empno) total\_emp, SUM(sal) total\_sal  FROM scott.emp  GROUP BY deptno;  --创建触发器  CREATE OR REPLACE TRIGGER EMP\_INFO   AFTER INSERT OR UPDATE OR DELETE ON scott.EMP  DECLARE   CURSOR CUR\_EMP IS    SELECT DEPTNO, COUNT(EMPNO) AS TOTAL\_EMP, SUM(SAL) AS TOTAL\_SAL FROM scott.EMP GROUP BY DEPTNO;  BEGIN   DELETE DEPT\_SAL; --触发时首先删除映射表信息   FOR V\_EMP IN CUR\_EMP LOOP    --DBMS\_OUTPUT.PUT\_LINE(v\_emp.deptno || v\_emp.total\_emp || v\_emp.total\_sal);    --插入数据    INSERT INTO DEPT\_SAL    VALUES     (V\_EMP.DEPTNO, V\_EMP.TOTAL\_EMP, V\_EMP.TOTAL\_SAL);   END LOOP;  END;  --对emp表进行DML操作  INSERT INTO emp(empno,deptno,sal) VALUES('123','10',10000);  SELECT \* FROM dept\_sal;  DELETE EMP WHERE empno=123;  SELECT \* FROM dept\_sal; |

显示结果如下：

4)、创建触发器，用来记录表的删除数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | --创建表  CREATE TABLE employee(    id  VARCHAR2(4) NOT NULL,    name VARCHAR2(15) NOT NULL,    age NUMBER(2)  NOT NULL,    sex CHAR NOT NULL  );  --插入数据  INSERT INTO employee VALUES('e101','zhao',23,'M');  INSERT INTO employee VALUES('e102','jian',21,'F');  --创建记录表(包含数据记录)  CREATE TABLE old\_employee AS SELECT \* FROM employee;  --创建触发器  CREATE OR REPLACE TRIGGER TIG\_OLD\_EMP   AFTER DELETE ON EMPLOYEE   FOR EACH ROW --语句级触发，即每一行触发一次  BEGIN   INSERT INTO OLD\_EMPLOYEE VALUES (:OLD.ID, :OLD.NAME, :OLD.AGE, :OLD.SEX); --:old代表旧值  END;  /  --下面进行测试  DELETE employee;  SELECT \* FROM old\_employee; |

5)、创建触发器，利用视图插入数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | --创建表  CREATE TABLE tab1 (tid NUMBER(4) PRIMARY KEY,tname VARCHAR2(20),tage NUMBER(2));  CREATE TABLE tab2 (tid NUMBER(4),ttel VARCHAR2(15),tadr VARCHAR2(30));  --插入数据  INSERT INTO tab1 VALUES(101,'zhao',22);  INSERT INTO tab1 VALUES(102,'yang',20);  INSERT INTO tab2 VALUES(101,'13761512841','AnHuiSuZhou');  INSERT INTO tab2 VALUES(102,'13563258514','AnHuiSuZhou');  --创建视图连接两张表  CREATE OR REPLACE VIEW tab\_view AS SELECT tab1.tid,tname,ttel,tadr FROM tab1,tab2 WHERE tab1.tid = tab2.tid;  --创建触发器  CREATE OR REPLACE TRIGGER TAB\_TRIGGER   INSTEAD OF INSERT ON TAB\_VIEW  BEGIN   INSERT INTO TAB1 (TID, TNAME) VALUES (:NEW.TID, :NEW.TNAME);   INSERT INTO TAB2 (TTEL, TADR) VALUES (:NEW.TTEL, :NEW.TADR);  END;  /  --现在就可以利用视图插入数据  INSERT INTO tab\_view VALUES(106,'ljq','13886681288','beijing');  --查询  SELECT \* FROM tab\_view;  SELECT \* FROM tab1;  SELECT \* FROM tab2; |

6)、创建触发器，比较emp表中更新的工资

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | --创建触发器  set serveroutput on;  CREATE OR REPLACE TRIGGER SAL\_EMP   BEFORE UPDATE ON EMP   FOR EACH ROW  BEGIN   IF :OLD.SAL > :NEW.SAL THEN    DBMS\_OUTPUT.PUT\_LINE('工资减少');   ELSIF :OLD.SAL < :NEW.SAL THEN    DBMS\_OUTPUT.PUT\_LINE('工资增加');   ELSE    DBMS\_OUTPUT.PUT\_LINE('工资未作任何变动');   END IF;   DBMS\_OUTPUT.PUT\_LINE('更新前工资 ：' || :OLD.SAL);   DBMS\_OUTPUT.PUT\_LINE('更新后工资 ：' || :NEW.SAL);  END;  /  --执行UPDATE查看效果  UPDATE emp SET sal = 3000 WHERE empno = '7788'; |

运行结果如下：

7)、创建触发器，将操作CREATE、DROP存储在log\_info表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | --创建表  CREATE TABLE log\_info(    manager\_user VARCHAR2(15),    manager\_date VARCHAR2(15),    manager\_type VARCHAR2(15),    obj\_name   VARCHAR2(15),    obj\_type   VARCHAR2(15)  );  --创建触发器  set serveroutput on;  CREATE OR REPLACE TRIGGER TRIG\_LOG\_INFO   AFTER CREATE OR DROP ON SCHEMA  BEGIN   INSERT INTO LOG\_INFO   VALUES    (USER,     SYSDATE,     SYS.DICTIONARY\_OBJ\_NAME,     SYS.DICTIONARY\_OBJ\_OWNER,     SYS.DICTIONARY\_OBJ\_TYPE);  END;  /  --测试语句  CREATE TABLE a(id NUMBER);  CREATE TYPE aa AS OBJECT(id NUMBER);  DROP TABLE a;  DROP TYPE aa;  --查看效果  SELECT \* FROM log\_info;  --相关数据字典-----------------------------------------------------  SELECT \* FROM USER\_TRIGGERS;  --必须以DBA身份登陆才能使用此数据字典  SELECT \* FROM ALL\_TRIGGERS;SELECT \* FROM DBA\_TRIGGERS;  --启用和禁用  ALTER TRIGGER trigger\_name DISABLE;  ALTER TRIGGER trigger\_name ENABLE; |

**四、Oracle存储过程**

**1、定义**所谓存储过程(Stored Procedure)，就是一组用于完成特定数据库功能的SQL语句集，该SQL语句集经过  
编译后存储在数据库系统中。在使用时候，用户通过指定已经定义的存储过程名字并给出相应的存储过程参数  
来调用并执行它，从而完成一个或一系列的数据库操作。

**2、存储过程的创建**Oracle存储过程包含三部分：过程声明，执行过程部分，存储过程异常。

（1）**无参存储过程语法**

create or replaceprocedureNoParPro

 as  //声明

 ;

 begin //执行

 ;

 exception//存储过程异常

 ;

 end;

（2）**带参存储过程实例**

create or replaceprocedure queryempname(sfindno emp.empno%type)

as

  sName emp.ename%type;

   sjobemp.job%type;

begin

      ....

exception

      ....

end;

（3）**带参数存储过程含赋值方式**

create or replaceprocedurerunbyparmeters

   (isal in emp.sal%type,

    sname out varchar,

    sjob in out varchar)

 as

   icount number;

 begin

      select count(\*) into icount from emp where sal>isal and job=sjob;

      if icount=**1** then

       ....

      else

      ....

     end if;

exception

     whentoo\_many\_rows then

    DBMS\_OUTPUT.PUT\_LINE('返回值多于1行');

     when others then

    DBMS\_OUTPUT.PUT\_LINE('在RUNBYPARMETERS过程中出错！');

end;

其中参数IN表示输入参数，是参数的默认模式。  
OUT表示返回值参数，类型可以使用任意Oracle中的合法类型。  
OUT模式定义的参数只能在过程体内部赋值，表示该参数可以将某个值传递回调用他的过程  
IN OUT表示该参数可以向该过程中传递值，也可以将某个值传出去。

（4）**存储过程中游标定义使用**

as //定义(游标一个可以遍历的结果集)

CURSOR cur\_1 IS

  SELECT area\_code,CMCODE,SUM(rmb\_amt)/**10000** rmb\_amt\_sn,

        SUM(usd\_amt)/**10000** usd\_amt\_sn

  FROM BGD\_AREA\_CM\_M\_BASE\_T

  WHERE ym >= vs\_ym\_sn\_beg

       AND ym <= vs\_ym\_sn\_end

  GROUP BY area\_code,CMCODE;

begin //执行（常用For语句遍历游标）

FOR rec IN cur\_1 LOOP

  UPDATE xxxxxxxxxxx\_T

   SET rmb\_amt\_sn= rec.rmb\_amt\_sn,usd\_amt\_sn = rec.usd\_amt\_sn

   WHERE area\_code = rec.area\_code

   AND CMCODE = rec.CMCODE

   AND ym = is\_ym;

END LOOP;

**（5）游标的定义**

--显示cursor的处理

declare

---声明cursor,创建和命名一个sql工作区

cursor cursor\_name is

    select real\_name from account\_hcz;

   v\_realname varchar2(**20**);

begin

    open cursor\_name;---打开cursor,执行sql语句产生的结果集

    fetch cursor\_name intov\_realname;--提取cursor,提取结果集中的记录

   dbms\_output.put\_line(v\_realname);

    closecursor\_name;--关闭cursor

end;

**3、在Oracle中对存储过程的调用**

（1）过程调用方式一

declare

     realsal emp.sal%type;

     realname varchar(**40**);

     realjob varchar(**40**);

begin   //过程调用开始

     realsal:=**1100**;

     realname:='';

     realjob:='CLERK';

      runbyparmeters(realsal,realname,realjob);－－必须按顺序

     DBMS\_OUTPUT.PUT\_LINE(REALNAME||'   '||REALJOB);

END;  //过程调用结束

（2）过程调用方式二

declare

    realsal emp.sal%type;

    realname varchar(**40**);

    realjob varchar(**40**);

begin    //过程调用开始

    realsal:=**1100**;

    realname:='';

    realjob:='CLERK';

     －－指定值对应变量顺序可变

    runbyparmeters(sname=>realname,isal=>realsal,sjob=>realjob);

   DBMS\_OUTPUT.PUT\_LINE(REALNAME||'   '||REALJOB);

END;  //过程调用结束

（3）过程调用方式三（SQL命令行方式下）

**1**、SQL>exec  proc\_emp('参数1','参数2');//无返回值过程调用

**2**、SQL>var vsal number

    SQL> exec proc\_emp ('参数1',:vsal);//有返回值过程调用

      或者：call proc\_emp ('参数1',:vsal);// 有返回值过程调用

**五、Oracle事务**

**1. 什么是事务**

在数据库中事务是工作的逻辑单元，一个事务是由一个或多个完成一组的相关行为的SQL语句组成，通过事务机制确保这一组SQL语句所作的操作要么都成功执行，完成整个工作单元操作，要么一个也不执行。

如：网上转帐就是典型的要用事务来处理，用以保证数据的一致性。

**2. 事务特性**

SQL92标准定义了数据库事务的四个特点：

* 原子性(Atomicity)：一个事务里面所有包含的SQL语句是一个执行整体，不可分割，要么都做，要么都不做。
* 一致性(Consistency)：事务开始时，数据库中的数据是一致的，事务结束时，数据库的数据也应该是一致的。
* 隔离性(Isolation)：是指数据库允许多个并发事务同时对其中的数据进行读写和修改的能力，隔离性可以防止事务的并发执行时，由于他们的操作命令交叉执行而导致的数据不一致状态。
* 持久性 (Durability) : 是指当事务结束后，它对数据库中的影响是永久的，即便系统遇到故障的情况下，数据也不会丢失。

一组SQL语句操作要成为事务，数据库管理系统必须保证这组操作的原子性（Atomicity）、一致性（consistency）、隔离性（Isolation）和持久性（Durability），这就是ACID特性。

**3. 数据异常**

因为Oracle中支持多个事务并发执行，所以会出现下面的数据异常。

**3.1 脏读**

当一个事务修改数据时，另一事务读取了该数据，但是第一个事务由于某种原因取消对数据修改，使数据返回了原状态，这是第二个事务读取的数据与数据库中数据不一致，这就叫脏读。

如：事务T1修改了一条数据，但是还未提交，事务T2恰好读取到了这条修改后了的数据，此时T1将事务回滚，这个时候T2读取到的数据就是脏数据。

**3.2 不可重复读(再次读取时发现数据改变，和上次的不一样)(update,delete)**

是指一个事务读取数据库中的数据后，另一个事务则更新了数据，当第一个事务再次读取其中的数据时，就会发现数据已经发生了改变，这就是不可重复读取。不可重复读取所导致的结果就是一个事务前后两次读取的数据不相同。

如：事务T1读取一行记录，紧接着事务T2修改了T1刚刚读取的记录，然后T1再次查询，发现与第一次读取的记录不同。

**3.3 幻读(再次读取时发现新增了记录，和上次的不一样，和不可重复读的区别)(insert)**

如果一个事务基于某个条件读取数据后，另一个事务则更新了同一个表中的数据，这时第一个事务再次读取数据时，根据搜索的条件返回了不同的行，这就是幻读。

如：事务T1读取一条指定where条件的语句，返回结果集。此时事务T2插入一行新记录，恰好满足T1的where条件。然后T1使用相同的条件再次查询，结果集中可以看到T2插入的记录，这条新纪录就是幻读。

事务中遇到的这些异常与事务的隔离性设置有关，事务的隔离性设置越多，异常就出现的越少，但并发效果就越低，事务的隔离性设置越少，异常出现的越多，并发效果越高。

**4. 事务隔离级别**

针对读取数据时可能产生的不一致现象，在SQL92标准中定义了4个事务的隔离级别：

| **隔离级别** | **脏读** | **不可重复读** | **幻读** |
| --- | --- | --- | --- |
| Read uncommitted(读未提交) | 是 | 是 | 是 |
| Read committed（读已提交） | 否 | 是 | 是 |
| Repeatable read（可重复读） | 否 | 否 | 是 |
| Serializable（串行读） | 否 | 否 | 否 |

Oracle默认的隔离级别是read committed。

Oracle支持上述四种隔离级别中的两种:read committed 和serializable。除此之外，Oralce中还定义Read only和Read write隔离级别。

Read only：事务中不能有任何修改数据库中数据的操作语句，是Serializable的一个子集。

Read write：它是默认设置，该选项表示在事务中可以有访问语句、修改语句，但不经常使用。

**设置隔离级别**

设置一个事务的隔离级别：

* SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
* SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
* SET TRANSACTION READ ONLY;
* SET TRANSACTION READ WRITE;

注意：这些语句是互斥的，不能同时设置两个或两个以上的选项。

设置单个会话的隔离级别：

* ALTER SESSION SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
* ALTER SESSION SET TRANSACTION ISOLATION SERIALIZABLE;

**5. 事务控制命令**

**5.1 提交事务**

在执行使用COMMIT语句可以提交事务，当执行了COMMIT语句后，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事务变化后的新数据。

**5.2 回滚事务**

保存点（savepoint）：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，通过指定保存点可以回退到指定的点。

设置保存点：

sql> Savepoint a;

删除保存点：

sql> Release Savepointa;

回滚部分事务：

sql> Rollback To a;

回滚全部事务：

sql> Rollback;

**6. 数据库锁**

数据库是一个多用户使用的共享资源。当多个用户并发地存取数据时，在数据库中就会产生多个事务同时存取同一数据的情况。若对并发操作不加控制就可能会读取和存储不正确的数据，破坏数据库的一致性。

在数据库中有两种基本的锁类型：排它锁（Exclusive Locks，即X锁）和共享锁（Share Locks，即S锁）。当数据对象被加上排它锁时，其他的事务不能对它读取和修改；加了共享锁的数据对象可以被其他事务读取，但不能修改。

**6.1 锁分类**

根据保护对象的不同，Oracle数据库锁可分为：

* DML lock（data locks，数据锁）：用于保护数据的完整性。
* DDL lock（dictionary locks，字典锁）：用于保护数据库对象的结构（例如表、视图、索引的结构定义）。
* Internal locks 和latches（内部锁与闩）：保护内部数据库结构。
* Distributed locks（分布式锁）：用于OPS（并行服务器）中。
* PCM locks（并行高速缓存管理锁）：用于OPS（并行服务器）中。

在Oracle中最主要的锁是DML锁，DML锁的目的在于保证并发情况下的数据完整性。在Oracle数据库中，DML锁主要包括TM锁和TX锁，其中TM锁称为表级锁，TX锁称为事务锁或行级锁。

锁出现在数据共享的场合，用来保证数据的一致性。当多个会话同时修改一个表时，需要对数据进行相应的锁定。

锁有“共享锁”、“排它锁”，“共享排它锁”等多种类型，而且每种类型又有“行级锁” (一次锁住一条记录)，“页级锁” (一次锁住一页，即数据库中存储记录的最小可分配单元)，“表级锁” (锁住整个表)。

**6.2 共享锁（S锁）**

可通过lock table in share mode命令添加该S锁。在该锁定模式下，不允许任何用户更新表。但是允许其他用户发出select …from for update命令对表添加RS锁。

**6.3 排他锁（X锁）**

可通过lock table in exclusive mode命令添加X锁。在该锁定模式下，其他用户不能对表进行任何的DML和DDL操作，该表上只能进行查询。

**6.4 行级共享锁（RS锁）**

通常是通过select … from for update语句添加的，同时该方法也是我们用来手工锁定某些记录的主要方法。比如，当我们在查询某些记录的过程中，不希望其他用户对查询的记录进行更新操作，则可以发出这样的语句。当数据使用完毕以后，直接发出rollback命令将锁定解除。当表上添加了RS锁定以后，不允许其他事务对相同的表添加排他锁，但是允许其他的事务通过DML语句或lock命令锁定相同表里的其他数据行。

**6.5 行级排他锁（RX锁）**

当进行DML操作时会自动在被更新的表上添加RX锁，或者也可以通过执行lock命令显式的在表上添加RX锁。在该锁定模式下，允许其他的事务通过DML语句修改相同表里的其他数据行，或通过lock命令对相同表添加RX锁定，但是不允许其他事务对相同的表添加排他锁（X锁）。

**6.6 共享行级排他锁（SRX锁）**

通过lock table in share row exclusive mode命令添加SRX锁。该锁定模式比行级排他锁和共享锁的级别都要高，这时不能对相同的表进行DML操作，也不能添加共享锁。

上述几种锁模式中，RS锁是限制最少的锁，X锁是限制最多的锁。它们的兼容关系如下：

基本上所有的锁都可以由Oracle内部自动创建和释放，但是其中的DDL和DML锁是可以通过命令进行管理的，命令语法：

LOCK table\_name IN

    [row share][row exclusive][share][share rowexclusive][exclusive] MODE

[NOWAIT];

下图列出产生锁定模式的SQL语句：

当程序对所做的修改进行提交(Commit)或回滚(Rollback)后，锁住的资源便会得到释放，从而允许其他用户进行操作。如果两个事务，分别锁定一部分数据，而都在等待对方释放锁才能完成事务操作，这种情况下就会发生死锁

**7. 数据库事务实现机制**

几乎所有的数据库管理系统中，事务管理的机制都是通过使用日志文件来实现的，我们来简单介绍一下日志的工作方式。

当用户执行一条修改数据库的DML语句时，DBMS自动在日志文件中写一条记录，显示被这条语句影响的每一条记录的两个副本。一个副本显示变化前的记录，另一个副本显示变化后的记录。当日志写完之后，DBMS才实际对磁盘中的记录进行修改。

如果用户随后执行COMMIT语句，事务结束也被记录在事务日志中。如果用户执行ROLLBACK语句，DBMS检查日志，找出自事务开始以来被修改的记录“以前”的样子，然后使用这些信息恢复它们以前的状态，有效地撤销事务期间对数据库所做的修改。

如果系统出错，系统操作员通常通过运行DBMS提供的特殊恢复程序来复原数据库。恢复程序检查到事务日志末尾，查找故障之前没有被提交的事务。恢复程序回滚没有完全完成的事务，以便仅有被提交的事务反映到数据库中，而故障中正处理的事务被回滚。

事务日志的使用明显增加了更新数据库的开销。在实际中，主流商用DBMS产品使用的日志技术比上述描述的方案更复杂，用以减小这种开销。此外，事务日志通常被存储在高速磁盘驱动器中，不同于存储数据库的磁盘，以减小磁盘访问竞争。某些个人计算机DBMS产品允许关闭事务日志性能，以提高DBMS的性能。

**8. 示例**

银行转帐的例子是最经典的事务示例：

用户把钱从一个银行账号转账至另一个银行账号，需要将资金从一个银行账号中取出，然后再存入另一个银行账号中。理想来说，这两次操作都应该成功。但是，如果有错误发生，则两次操作都应该失败，否则的话，操作之后其中一个账号中的金额将会是错误的，整个操作过程应该是原子性的，两个操作都是一个原子事务操作的一部分。

示例：

-- 从账户一向账户二转账

DECLARE

 v\_money NUMBER(**8**, **2**); -- 转账金额

  v\_balanceaccount.balance%TYPE; --账户余额

BEGIN

 v\_money := &转账金额; -- 输入转账金额

  -- 从账户一减钱

  UPDATE account SETbalance = balance -v\_money WHERE id=&转出账户

 RETURNING balance INTO v\_balance;

  IF SQL%NOTFOUND THEN

   RAISE\_APPLICATION\_ERROR(-**20001**, '没有该账户：'||&转出账户);

  END IF;

  IF v\_balance < **0** THEN

   RAISE\_APPLICATION\_ERROR(-**20002**, '账户余额不足');

  END IF;

  -- 向账户二加钱

  UPDATE account SETbalance = balance +v\_money WHERE id=&转入账户;

  IF SQL%NOTFOUND THEN

   RAISE\_APPLICATION\_ERROR(-**20001**, '没有该账户：'||&转入账户);

  END IF;

  -- 如果没有异常，则提交事务

  COMMIT;

 DBMS\_OUTPUT.PUT\_LINE('转账成功');

 EXCEPTION

    WHEN OTHERS THEN

      ROLLBACK; -- 出现异常则回滚事务

     DBMS\_OUTPUT.PUT\_LINE('转账失败：');

      DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

**六、Oracle锁**

以前虽然在网上看到很多关于Oracle锁机制的描述，但总感觉哪里有缺陷不适合自己，因此花了点时间参考Tom Tyke的《Oracle 9i/10g/11g编程艺术》一书整理了一下Oracle锁相关的知识。

**一、Oracle数据库的锁类型：**

根据保护的对象不同，Oracle数据库锁可以分为以下几大类：

1、DML锁（data locks，数据锁），用于保护数据的完整性；

2、DDL锁（dictionary locks，字典锁），用于保护数据库对象的结构，如表、索引等的结构定义；

3、内部锁和闩（internal locks and latches），保护数据库的内部结构。

**二、接下来依次讨论以上三种锁结构：**

**1.DML锁**

DML锁主要包括TM锁和TX锁，其中TM锁称为意向锁或表级锁，TX锁称为行级锁或事务锁。我们可以认为Oracle只有如下6种LMODE的锁，只是根据锁定的对象不同而有不同的名称，如6号的X锁，既可以是用于锁表的TM锁，也可以是TX锁，也可以是DDL锁。

**1.1 TM锁（也叫意向锁/表级锁）**

TM锁包含如下类型：

TM锁的兼容性如下：（Y表示兼容，N表示冲突）

**1.2 TX锁**

TX的本义是Transaction（事务），当一个事务第一次执行数据更改（Insert、Update、Delete）或使用SELECT… FOR UPDATE语句进行查询时，它即获得一个TX（事务）锁，直至该事务结束（执行COMMIT或ROLLBACK操作）时，该锁才被释放。

在同一个事务中，无论是锁定一行，还是一百万行，对于Oracle来说TX锁的开销是一样的。因此Oracle从来都不会锁升级，因为事务锁只含表级锁和行级锁，而行级锁的开销是不随锁定的行数变化的。这点可能与其他数据库不一样，原因是针对Oracle的每行数据，都有一个标志位来表示该行数据是否被锁定。这样就极大的减小了行级锁的维护开销，也不可能出现锁升级。数据行上的锁标志一旦被置位，就表明该行数据被加X锁。

Oracle在数据行上没有S锁，换句话说就是TX锁只有一种--行级独占锁**。**（注意TX锁在v$lock的lmode也是6，但是这个6与TM锁的6号X锁只是因为锁定的对象不同而被叫做了TX锁）

**1.3 举例说明**

**当发出一个DML命令后会话获取一个3号的TM锁，和一个针对特定行的6号TX锁。**

行级只有X锁，且锁模式为6，再次重申这里的6并不是指TM的6号表锁。此外Oracle一个事务中无论锁定多少行只会获取一个TX锁，这点上边已经解释过了，但有多少个表对象就会获取多少个TM表级锁。

验证如下：

查询锁的语句为：

|  |  |
| --- | --- |
| 1 | select sid,type,id1,lmode,request,block from v$lock l where sid in (select session\_id from v$locked\_object) and type in ('TM', 'TX') order by 1; |

**1.4 DML锁的总结：**

读永远不会阻止写，因为读只加NULL锁。但有唯一的一个例外，就是select ...for update。

写永远不会阻塞读（默认隔离级别下），因为一致性读的存在，相关原理可以到网上搜索Oracle一致性读的实现，Oracle会通过回滚段（undo）提供给数据的一致性读。

**注意：以上说明的读和写不会互相阻塞是指在事务锁（TM）级别不会，但读写之间依然会发生数据库内部闩锁的争用。具体可以参考数据库内部闩锁的博文。**

**2.DDL锁**

**重点：DDL是保护表结构定义的。**

当DDL命令发出时，Oracle会自动在被处理的对象上添加DDL锁定，从而防止对象被其他用户所修改。当DDL命令结束以后，则释放DDL锁定。DDL锁定不能显式的被请求，只有当对象结构被修改或者被引用时，才会在对象上添加DDL锁定。

并不是所有DDL都会触发DDL锁，例如现在的创建索引语句，就只会获取一个S模式的TM锁，因此不会阻塞读。而online模式创建索引的语句则只会获取一个RS模式的TM锁，因此连DML也不会被阻塞。

需要注意的是DDL总会提交，即便是执行不成功也是如此，因此如果在事务中执行了DDL语句会导致所有事物被提交。验证很容易，在一个窗口执行一条delete然后执行DDL,你会发现记录被不可逆转的删除了，RollBack无效。因此针对事务中的DDL请务必使用自治事务实现。

**DDL锁有3种：(第一种在表对象上的体现就是X模式的TM锁)**

**2.1 排他DDL锁**

一般对表的DDL语句都会获取一个X模式的TM锁，这是为什么在表结构更改时只能查询不能修改的原因。

**2.2 共享DDL锁**

共享DDL锁的常见情形为创建存储过程时，会尝试为所有涉及到的表添加共享DDL锁，这会允许类似的DDL操作并发，但会阻止所有想要获取排他DDL锁的会话（即更改表结构的会话）。

可以认为这就是4号TM表锁。

**2.3 可中断解析锁**

会话解析一条语句时，对于该语句引用的每一个对象都会施加解析锁，这个目的是如果以某种方式删除或修改了引用对象，可以将共享池中已经解析的无效缓存语句刷出。

**七、Oracle包**

1.Oracle中的包和包体

Oracle中的包和包体与java中的接口和类才关系特别类似，我们就根据对比学习一下包和包体吧！

2.oracle包和包体与自定义函数，过程区别

2.1 如果直接create 函数，函数不会出现在包里，而是在function目录下面，如果在包里创建，则会出现在包里，他们两者有什么区别？

**答：**

|  |  |
| --- | --- |
| 1  2  3  4 | 直接创建的函数，是数据公共函数。在调用函数的时候直接调用函数名带参数就可以。而建在包体里的函数是私有函数，在有在包体里才能直接用函数名带参数调用。  外部程序要调用需要--包名.函数名（参数）--这样去调用。 |

3.创建Oracle包以及实现包

1.Oracle中的包和包体

Oracle中的包和包体与java中的接口和类才关系特别类似，我们就根据对比学习一下包和包体吧！

2.oracle包和包体与自定义函数，过程区别

2.1 如果直接create 函数，函数不会出现在包里，而是在function目录下面，如果在包里创建，则会出现在包里，他们两者有什么区别？

**答：**

|  |  |
| --- | --- |
| 1  2  3  4 | 1）直接创建的函数，是数据公共函数。在调用函数的时候直接调用函数名带上参数就可以。    2）而建在包体里的函数是私有函数，在有在包体里才能直接用函数名带参数调用。  外部程序要调用需要--包名.函数名（参数）--这样去调用。 |

3.创建Oracle包以及实现包

**sql语句如下：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | -- 包(pakage)和包体(package body)  -- 首先创建一个包含字段comm的emp表，再创建下面的包和包体    create or replace package test\_package  is      -- 声明全局变量：默认奖金数额      g\_comm number:=100;      -- 声明存储过程：用于重置奖金数额      procedure proc\_reset\_comm(p\_comm in number);      -- 声明函数：用于求所有员工中最高工资      function func\_maxsal\_emp return number;  end test\_package;  /    create or replace package body test\_package  is      -- 实现存储过程      procedure proc\_reset\_comm(p\_comm in number)      is      begin          if nvl(p\_comm,0)>g\_comm then              g\_comm:=p\_comm;          end if;          dbms\_output.put\_line(g\_comm);      end proc\_reset\_comm;      -- 实现函数      function func\_maxsal\_emp return number      is      v\_maxsal emp.sal%type;      begin          select max(sal) into v\_maxsal from a\_emp;          return v\_maxsal;      end func\_maxsal\_emp;  end;  /    -- 测试包和包体  exec test\_package.proc\_reset\_comm(111);  select test\_package.func\_maxsal\_emp() from dual; |

**八、Oracle游标**

游标用来处理从数据库中检索的多行记录（使用SELECT语句）。利用游标，程序可以逐个地处理和遍历一次检索返回的整个记录集。

     为了处理SQL语句，Oracle将在内存中分配一个区域，这就是上下文区。这个区包含了已经处理完的行数、指向被分析语句的指针，整个区是查询语句返回的数据行集。游标就是指向上下文区句柄或指针。

两种游标：  
显示游标（需要明确定义！）

    显示游标被用于处理返回多行数据的SELECT 语句，游标名通过CURSOR….IS 语句显示地赋给SELECT语句。

       在PL/SQL中处理显示游标所必需的四个步骤：

       1）声明游标；CURSOR cursor\_name IS select\_statement

       2）为查询打开游标；OPEN cursor\_name

       3）取得结果放入PL/SQL变量中；

             FETCH cursor\_name INTO list\_of\_variables;

             FETCH cursor\_name INTO PL/SQL\_record;

       4）关闭游标。CLOSE cursor\_name

*注意：在声明游标时，*select\_statement不能包含INTO子句。当使用显示游标时，INTO子句是FETCH语句的一部分。  
   1、 显式游标  
      select语句上 使用显式游标  
      能明确访问结果集  
         for循环游标  
         参数游标   
           解决多行记录的查询问题  
         fetch游标  
隐式游标   
       所有的隐式游标都被假设为只返回一条记录。  
      使用隐式游标时，用户无需进行声明、打开及关闭。PL/SQL隐含地打开、处理，然后关掉游标。  
      例如：  
      …….  
       SELECT studentNo,studentName

       INTO curStudentNo,curStudentName

       FROM StudentRecord

       WHERE name=’gg’;

       上述游标自动打开，并把相关值赋给对应变量，然后关闭。执行完后，PL/SQL变量curStudentNo,curStudentName中已经有了值。  
   2、隐式游标  
     单条sql语句所产生的结果集合   
       用关键字SQL表示隐式游标  
        4个属性%rowcount  影响的记录的行数  整数  
               %found     影响到了记录 true  
               %notfound  没有影响到记录 true  
               %isopen    是否打开  布尔值 永远是false  
         多条sql语句隐式游标SQL永远指的是最后一条sql语句的结果  
         主要使用在update和 delete语句上       
  
  
**实际操作和例子：**

  (1)FOR循环游标 (常用的一种游标)

 --<1>定义游标  
 --<2>定义游标变量  
 --<3>使用for循环来使用这个游标

  --前向游标只能往一个方向走  
  --效率很高  
      declare  
        --类型定义  
        cursor cc is selectempno,ename,job,sal  
         from emp where job ='MANAGER';  
        --定义一个游标变量  
        ccrec cc%rowtype;  
      begin  
         --for循环  
        for ccrec in cc loop  
           dbms\_output.put\_line(ccrec.empno||'-'||ccrec.ename||'-'||ccrec.job||'-'||ccrec.sal);  
         end loop;         
      end;  
   (2) fetch游标  
     --使用的时候 必须要明确的打开和关闭  
      declare  
        --类型定义  
        cursor cc is selectempno,ename,job,sal  
         from emp where job ='MANAGER';  
        --定义一个游标变量  
        ccrec cc%rowtype;  
      begin  
        --打开游标  
         open cc;  
        --loop循环  
         loop  
            --提取一行数据到ccrec中   
            fetch ccinto ccrec;           
            --判断是否提取到值，没取到值就退出  
            --取到值cc%notfound 是false  
            --取不到值cc%notfound 是true   
            exit whencc%notfound;  
           dbms\_output.put\_line(ccrec.empno||'-'||ccrec.ename||'-'||ccrec.job||'-'||ccrec.sal);              
         end loop;   
        --关闭游标  
         close cc;    
      end;  
  游标的属性4种  
       %notfound  fetch是否提到数据没有true 提到false  
       %found      fetch是否提到数据有true 没提到false  
       %rowcount  已经取出的记录的条数  
       %isopen    布尔值游标是否打开  
   (3)参数游标  
 按部门编号的顺序输出部门经理的名字  
     declare  
       --部门  
       cursor c1 is select deptno from dept;  
       --参数游标c2,定义参数的时候  
       --只能指定类型,不能指定长度    
       --参数只能出现在select语句=号的右侧  
       cursor c2(no number,pjob varchar2) isselect emp.\* from emp  
         where deptno = no andjob=pjob;  
       c1rec c1%rowtype;  
       c2rec c2%rowtype;  
       --定义变量的时候要指定长度  
       v\_job varchar2(20);  
     begin  
       --部门  
        for c1rec in c1 loop  
          --参数在游标中使用  
          for c2rec inc2(c1rec.deptno,'MANAGER') loop  
           dbms\_output.put\_line(c1rec.deptno||'-'||c2rec.ename);  
          end loop;   
        end loop;   
     end;   
  (4)引用游标/动态游标  
       --select语句是动态的  
     declare  
       --定义一个类型(ref cursor)弱类型      
       type cur is ref cursor;  
         --强类型(返回的结果集有要求)  
       type cur1 is ref cursor returnemp%rowtype;  
       --定义一个ref cursor类型的变量     
       cura cur;  
       c1rec emp%rowtype;  
       c2rec dept%rowtype;  
     begin  
  DBMS\_output.put\_line('输出员工')  ;         
       open cura for select \* from emp;  
       loop  
         fetch cura intoc1rec;     
         exit when cura%notfound;  
        DBMS\_output.put\_line(c1rec.ename)   ;  
       end loop ;  
  DBMS\_output.put\_line('输出部门')   ;  
       open cura for select \* from dept;  
       loop  
         fetch cura intoc2rec;     
         exit when cura%notfound;  
        DBMS\_output.put\_line(c2rec.dname)   ;  
       end loop;    
       close cura;  
    end;

**九、Oracle函数**

**F.1字符函数——返回字符值**

(chr,concat,initcap,lower,lpad/rpad,nls\_initcap,nls\_lower,nls\_upper,regexp\_replace,regexp\_substr,replace,trim/ltrim/rtrim,soundex,substr,translate,upper)

说明：可以sql和plsql中使用

CHR

语法:  chr(x)

功能：给出整数X，返回对应的ASCII码字符。CHR和ASCII是一对反函数。

SQL> select chr(54740) 别名1,chr(65) 别名2 from dual;

别名1  别名2

赵      A

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

**CONCAT**

语法： CONCAT（string1,string2）

功能：连接两个字符串

SQL> select concat('010-','88888888')||'23'  连接 from dual;

连接

010-8888888823

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

**INITCAP**

语法：INITCAP（string）

功能：返回字符单词首字母大写，其余小写，单词用空格和非字母字符分隔。

SQL> select initcap('smith hEllo') upp from dual;

UPP

Smith Hello

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

**LOWER**

    语法：LOWER(string)

     功能：所以字母小写

     SQL> select lower('AaBbCcDd') AaBbCcDd fromdual;

     AaBbCcDd

     aabbccdd

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

**LPAD/RPAD**

语法：LPAD/RPAD（string1,x[,string2]）

功能：在string1字符左边或右边粘贴数个string2字符，直到字符总字节数达到x字节。string2默认为空格。

如果string2的长度要比X字符少，就按照需要进行复制。如果string2多于X字符，则仅string2前面的X各字符被使用。如果string1长度大于x,则返回string1左端x个字符。

RPAD  在列的右边粘贴字符

LPAD  在列的左边粘贴字符

SQL> select lpad(rpad('gao',10,'\*'),17,'\*')from dual;

LPAD(RPAD('GAO',1

\*\*\*\*\*\*\*gao\*\*\*\*\*\*\*

不够字符则用\*来填满

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

**NLS\_INITCAP**

语法：NLS\_INITCAP（string[,nlsparams]）

功能：返回字符串每个单词第一个字母大写而单词中的其他字母小写的string，nlsparams

指定了不同于该会话缺省值的不同排序序列。如果不指定参数，则功能和INITCAP相同。Nlsparams可以使用的形式是：‘NLS\_SORT=sort’ 这里sort制订了一个语言排序序列。

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

**NLS\_LOWER**

语法：NLS\_LOWER（string[,nlsparams]）

功能：返回字符串中的所有字母都是小写形式的string。不是字母的字符不变。

Nlsparams参数的形式与用途和NLS\_INITCAP中的nlsparams参数是相同的。如果nlsparams没有被包含，那么NLS\_LOWER所作的处理和LOWER相同。

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

**NLS\_UPPER**

语法：NLS\_UPPER（string[,nlsparams]）

功能：返回字符串中的所有字母都是大写的形式的string。不是字母的字符不变。nlsparams参数的形式与用途和NLS\_INITCAP中的相同。如果没有设定参数，则NLS\_UPPER功能和UPPER相同。

使用位置：过程性语句和SQL语句。

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

**REGEXP\_REPLACE**

语法：REGEXP\_REPLACE(str1,pattem[,str2[,pos[,occ[,par]]]])

功能：10g新增函数，扩展了REPLACE函数的功能，并且用于按照特定正则表达式的规则替换字符串。其中参数str1指定源字符表达式，pattem指定正则表达式，str2指定替换字符串，pos指定起始搜索位置，occ指定替换出现的第几个字符串，par指定默认匹配操作的文本串。

select REGEXP\_REPLACE(a,’(.)’,’\1’) a from count;

A r g e n t i n a

体会NVL为DECODE，只支持NVL()内不再有其它括号()

select a,

       instr(upper(a), 'NVL(', 1) a3,

       instr(upper(a), ')',instr(upper(a),'NVL(', 1),1) a4,

       substr(a,instr(upper(a), 'NVL(',1),instr(upper(a), ')',instr(upper(a), 'NVL(', 1),1)-instr(upper(a), 'NVL(',1)+1) a41,

       substr(a,instr(upper(a), 'NVL(',1)+4,instr(upper(a), ')',instr(upper(a), 'NVL(', 1), 1)-instr(upper(a), 'NVL(',1)-4) a5,

       REGEXP\_REPLACE(

       substr(a,instr(upper(a), 'NVL(',1)+4,instr(upper(a), ')',instr(upper(a), 'NVL(', 1), 1)-instr(upper(a), 'NVL(',1)-4),

       '(.\*),(.\*)','\2,\1'

       ) a6,

       REGEXP\_REPLACE(

       substr(a,instr(upper(a), 'NVL(',1)+4,instr(upper(a), ')',instr(upper(a), 'NVL(', 1), 1)-instr(upper(a), 'NVL(',1)-4),

      '(.\*),(.\*)','decode(\1,null,\2,'''',\2,\1)'

       ) a7,

       substr(a,1,instr(upper(a), 'NVL(',1)-1)||REGEXP\_REPLACE(

       substr(a,instr(upper(a), 'NVL(',1)+4,instr(upper(a), ')',instr(upper(a), 'NVL(', 1), 1)-instr(upper(a), 'NVL(',1)-4),

      '(.\*),(.\*)','decode(\1,null,\2,'''',\2,\1)'

       )||substr(a,instr(upper(a), ')',instr(upper(a),'NVL(', 1), 1)+1) a8

  from temp\_liut a;

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

判断是否是数字

regexp\_replace(a, '\d+', '') is null

**REGEXP\_SUBSTR**

语法：REGEXP\_SUBSTR(str1,pattem [,pos[,occ[,par]]])

功能：10g新增函数，扩展了SUBSTR函数的功能，并且用于按照特定表达式的规则返回字符串的子串。其中参数str1指定源字符表达式，pattem指定规则表达式， pos指定起始搜索位置，occ指定替换出现的第几个字符串，par指定默认匹配操作的文本串。

**REPLACE**

语法：REPLACE（string，search\_str[,replace\_str]）

功能：把string中的所有的子字符串search\_str用可选的replace\_str替换，如果没有指定replace\_str，所有的string中的子字符串search\_str都将被删除。REPLACE是TRANSLATE所提供的功能的一个子集。

REPLACE('string','s1','s2')

string   希望被替换的字符或变量

s1       被替换的字符串

s2       要替换的字符串

SQL> select replace('he lohe you','he','i') from dual;

replace('he lohe you','he','i')

i loi you

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

**TRIM/LTRIM/RTRIM**

语法1：LTRIM/RTRIM（string1,[string2]）

语法2：trim([string2] from string1)

语法1功能：中删除从左/右边算起出现在string1中的字符string2，string2如果是多个字符则逐个单字符比对删除,tring2被缺省设置为单个的空格。当遇到不在string2中的第一个字符，结果就被返回了；

语法2功能：删除左右两边出现在string1中的字符string2，tring2必须为单字符，否则报错。

select ltrim(rtrim('   gao qian jing   ',' '),' ')from dual;

gao qian jing

select ltrim('abaaaabbbcda','ab') from dual;

cda

select trim('a' from 'abacda') from dual;

bacd

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

**SOUNDEX**

语法:  SOUNDEX（string）

功能:  返回string的声音表示形式.这对于比较两个拼写不同但是发音类似的单词而言很有帮助，如果字符发音相同，则返回的结果会一致.

SOUNDEX 返回一个与给定的字符串读音相同的字符串

SQL> create table table1(xm varchar(8));

SQL> insert into table1 values('weather');

SQL> insert into table1 values('wether');

SQL> insert into table1 values('gao');

SQL> select xm from table1 where soundex(xm)=soundex('weather');

XM

weather

wether

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

**SUBSTR**

语法:  SUBSTR（string,a[,b]）

功能：截取字符串，从第a个开始取b个字符，这个务必要注意，是字符。 vachar2最长4000个字节，GBK编码中一个中文字符占2个字节，韩文字符占4个字节，如果string是date或者number的数据类型，会自动转化为varchar2。

SQL> select substr('13088888888',3,8) 截取字符串 from dual;

截取字符串

08888888

select SUBSTR(t.a,4),a from temp\_liut t;

JAN-00  04-jan-00

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

**TRANSLATE**

语法:  TRANSLATE(string,from\_str,to\_str)

功能:  将字符string按照from\_str与to\_str的对应规则进行处理，返回将所出现的from\_str中的每个字符替换为to\_str中的相应字符以后的string. TRANSLATE是REPLACE所提供的功能的一个超集.如果from\_str比to\_str长,那么在from\_str中而不在to\_str中而外的字符将从string中被删除,因为它们没有相应的替换字符. to\_str不能为空.Oracle把空字符串认为是NULL,并且如果TRANSLATE中的任何参数为NULL,那么结果也是NULL.

Select TRANSLATE('2abc2234','01234abcde','99999XXXXX') tra from dual

9XXX9999

select replace(TRANSLATE('as中国fd1234','1234567890','0000000000'),'0') from dual;

查找字符串',01234,2342,2,'中逗号出现次数

select length(translate(',01234,2342,2,', 'a0123456789', ' ')) from dual;

判断字符串是否是数字

replace(translate(a, '0123456789', '0'),'0') is null

regexp\_replace(a, '\d+', '') is null

**UPPER**

语法: UPPER（string）

功能: 所有字母大写.（不是字母的字符不变.如果string是CHAR数据类型的,那么结果也是CHAR类型的.如果string是VARCHAR2类型的,那么结果也是VARCHAR2类型的）.

 SQL> select upper('AaBbCcDd') upper from dual;

UPPER

AABBCCDD

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

**F.2    字符函数——返回数字**

(ascii,instr,instrb,length,lengthb,nls\_sort)

说明：可以sql和plsql中使用

**ASCII**

语法:  ASCII（string）

功能: 返回string字符串首字符的十进制表示ascii码值。 CHR和ASCII是互为相反的函数.CHR得到给定字符编码的响应字符. ASCII得到给定字符的字符编码.

SQL> select ascii('A') A,ascii('a') a,ascii('0') zero,ascii(' ') spacefrom dual;

A        A      ZERO     SPACE

65       97       48        32

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

**INSTR**

语法:  INSTR（str1, str2[,a,b]）

功能:  得到在str1中包含str2的位置. a>0,str1时从左边开始检查的,开始的位置为a;a<0,那么str1是从右边开始进行扫描的,开始的位置为a。第b次出现的位置将被返回. a和b都缺省设置为1,这将会返回在string1中第一次出现string2的位置.如果string2在a和b的规定下没有找到,那么返回0.位置的计算是相对于string1的开始位置的,不管a和b的取值是多少.

INSTR(C1,C2,I,J) 在一个字符串中搜索指定的字符,返回发现指定的字符的位置;

C1    被搜索的字符串

C2    希望搜索的字符串

I     搜索的开始位置,默认为1(如果为负数会从后向前搜索)

J     出现的位置,默认为1

SQL> select instr('oracle traning','ra',1,2) instring from dual;

INSTRING

9

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

**INSTRB**

语法:  INSTRB（string1, string2[a,[b]]）

功能:  和INSTR相同,只是操作的对参数字符使用的位置的是字节.

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

**LENGTH**

语法:  LENGTH（string）

功能:  返回字符串的长度，特别注意的，对于空的字段，返回为空，而不是0。

SELECT   LENGTH (' 130 ') 返回字符串长度  FROM DUAL;

返回字符串长度

5

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

**LENGTHB**

语法:  LENGTHB（string）

功能:  返回以字节为单位的string的长度.对于单字节字符集LENGTHB和LENGTH是一样的.

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

**NLS\_SORT**

语法: NLS\_SORT（string[,nlsparams]）

功能: 得到用于排序string的字符串字节.所有的数值都被转换为字节字符串,这样在不同数据库之间就保持了一致性. Nlsparams的作用和NLS\_INITCAP中的相同.如果忽略参数,会话使用缺省排序.

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

**F.3    数学函数**

(abs,acos,asin,atan,atan2,ceil,cos,cosh,exp,floor,ln,log,mod,power,round,sign,sin,sinh,sqrt,tan,tanh,trunc)

说明：数学函数的输入和输出都是数字型，并且多数函数精确到38位。函数cos\cosh\exp\ln\log\sin\sinh\sqrt\tan\tanh精确到36位，acos\asin\atan\atan2精确到30为。数学函数可以在sql语句和plsql块中引用。

**ABS**

语法:   ABS(x)

功能:   得到x的绝对值.

SQL> select abs(100),abs(-100) from dual;

ABS(100) ABS(-100)

100       100

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

**ACOS**

语法:  ACOS(x)

功能:  返回x的反余弦值. 输入x应该从-1到1之间的数,结果在0到pi之间,输出以弧度为单位.

SQL> select acos(-1) from dual;

ACOS(-1)

3.1415927

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

**ASIN**

语法:  ASIN(x)

功能:  返回x的反正弦值. X的范围应该是－1到1之间,返回的结果在－pi/2到pi/2之间,以弧度为单位.

SQL> select asin(0.5) from dual;

ASIN(0.5)

.52359878

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

**ATAN**

语法:  ATAN(x)

功能:  计算x的反正切值.返回值在－pi/2到pi/2之间,单位是弧度.

SQL> select atan(1) from dual;

ATAN(1)

.78539816

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

**ATAN2**

语法:  ATAN2(x,y)

功能: 返回x除以y的反正切值.结果在负的pi/2到正的pi/2之间,单位是弧度.

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

**CEIL**

语法:  CEIL(x)

功能:  计算大于或等于x的最小整数值.

SQL> select ceil(3.1415927) from dual;

CEIL(3.1415927)

      4

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

**COS**

语法:  COS(x)

功能:  返回x的余弦值. x的单位是弧度.

SQL> select cos(-3.1415927) from dual;

COS(-3.1415927)

     -1

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

**COSH**

语法:  COSH(x)

功能:  计算x的双曲余弦值.

SQL> select cosh(20) from dual;

COSH(20)

242582598

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

**EXP**

语法:  EXP(x)

功能:  计算e的x次幂. e为自然对数,约等于2.71828.

SQL> select exp(2),exp(1) from dual;

EXP(2)    EXP(1)

7.3890561 2.7182818

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

**FLOOR**

语法:  FLOOR(x)

功能:  返回小于等于x的最大整数值.

SQL> SELECT   FLOOR (2345.67), FLOOR (-2345.67) FROM dual;

FLOOR(2345.67)   FLOOR (-2345.67)

 2345                       -2346

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

**LN**

语法:  LN(x)

功能:  返回x的自然对数. x必须是正数,并且大于0

SQL> select ln(1),ln(2),ln(2.7182818) from dual;

LN(1)     LN(2) LN(2.7182818)

0 .69314718     .99999999

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

**LOG**

语法:  LOG(x,y)

功能:  计算以x为底的y的对数.底必须大于0而且不等于1, y为任意正数.

SQL> select log(2,1),log(2,4) from dual;

LOG(2,1)  LOG(2,4)

0         2

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

**MOD**

语法:  MOD(x,y)

功能:  返回x除以y的余数.如果y是0,则返回x

SQL> select mod(10,3),mod(3,3),mod(2,3) from dual;

MOD(10,3)  MOD(3,3)  MOD(2,3)

1        0         2

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

**POWER**

语法:  POWER(x,y)

功能:  计算x的y次幂.

POWER 返回n1的n2次方根

SQL> select power(2,10),power(3,3) from dual;

POWER(2,10) POWER(3,3)

1024         27

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

**ROUND**

语法:  ROUND(x[,y])

功能:  四舍五入函数，y缺省值为0，x保留整数；y>0，x保留小数点右边y位；y<0，x保留小数点左边 |y| 位;可以对时间进行round，效果是只保留年月日。

SELECT   ROUND (55.655, 2),   --55.66

         ROUND (55.654,2),   --55.65

         ROUND (45.654, -1), --50

         ROUND (45.654, -2), --0

         ROUND (55.654,-2)   --100

  FROM   DUAL;

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

**SIGN**

语法:  SIGN(x)

功能:  检测x的正负.如果x<0返回－1.如果x=0返回0.如果x>0返回1.

SQL> select sign(123),sign(-100),sign(0) from dual;

SIGN(123) SIGN(-100)   SIGN(0)

1        -1         0

常和decode 结合使用

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

**SIN**

语法:SIN(x)

功能:计算x的正弦值. X是一个以弧度表示的角度.

SQL> select sin(1.57079) from dual;

SIN(1.57079)

   1

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

**SINH**

语法:SINH(x)

功能:返回x的双曲正弦值.

SQL> select sin(20),sinh(20) from dual;

SIN(20)  SINH(20)

.91294525 242582598

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

**SQRT**

语法:  SQRT(x)

功能:  返回x的平方根. x必须是正数.

SQL> select sqrt(64),sqrt(10) from dual;

SQRT(64)  SQRT(10)

8 3.1622777

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

**TAN**

语法:  TAN(x)

功能:  计算x的正切值, x是一个以弧度位单位的角度.

SQL> select tan(20),tan(10) from dual;

TAN(20)   TAN(10)

2.2371609 .64836083

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

**TANH**

语法:  TANH(x)

功能:  计算x的双曲正切值.

SQL> select tanh(20),tan(20) from dual;

TANH(20)   TAN(20)

1 2.2371609

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

**TRUNC**

语法:  TRUNC(x[,y])

功能:  截取数字函数，只舍不入函数， y缺省值为0，x保留整数；y>0，x保留小数点右边y位；y<0，x保留小数点左边 |y| 位

SELECT   TRUNC (55.655, 2),   --55.65

         TRUNC (55.654,2),   --55.65

         TRUNC (45.654, -1), --40

         TRUNC (45.654, -2), --0

         TRUNC (55.654,-2)   --0

  FROM   DUAL;

SELECT   TRUNC (SYSDATE, 'DD'),  --当天

         TRUNC (SYSDATE,'MM'),  --本月第一天

         TRUNC (SYSDATE,'yyyy'),  --本年第一天

         TRUNC (SYSDATE,'day'),  --本周第一天

         TRUNC (SYSDATE,'q')  --本季度第一天

  FROM   DUAL;

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

**F.4  日期时间函数**

(add\_months,current\_date,current\_timestamp,dbtimesone,extract,from\_tz,last\_day,months\_between,new\_time,next\_day,numtodsinternal,numtoyminternal,round,sys\_extract\_utc,sysdate,systimestamp,to\_dsinternal,to\_timestamp,to\_timestamp\_tz,to\_yminternal,trunc,tz\_offset)

说明：日期时间函数用于处理date和timestamp类型的数据，除了函数months\_between返回数字外，其余均返回date类型，Oracle以7位数字格式来存放日期数据，包括世纪、年、月、日、小时、分钟、秒，并且默认日期显式格式为“DD-MON-YY”。

**ADD\_MONTHS**

语法:ADD\_MONTHS(d,x)

功能:返回日期d加上x个月后的月份。x可以是任意整数。如果结果日期中的月份所包含的天数比d日期中的“日”分量要少。（即相加后的结果日期中的日分量信息已经超过该月的最后一天，例如，8月31日加上一个月之后得到9月31日，而9月只能有30天）返回结果月份的最后一天。

使用位置:过程性语言和SQL语句。

SQL> select to\_char(add\_months(to\_date('199912','yyyymm'),2),'yyyymm')from dual;

TO\_CHA

200002

SQL> select to\_char(add\_months(to\_date('199912','yyyymm'),-2),'yyyymm')from dual;

TO\_CHA

199910

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

**CURRENT\_DATE**

语法: CURRENT\_DATE

功能:9i新增函数，返回当前会话时区所对应的日期时间。

select CURRENT\_DATE from dual;

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

**CURRENT\_TIMESTAMP**

语法:CURRENT\_TIMESTAMP

功能:9i新增函数，返回当前会话时区所对应的日期时间。

select CURRENT\_TIMESTAMP from dual;

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

**DBTIMESONE**

语法:DBTIMESONE

功能:9i新增函数，返回数据库所在时区。

select DBTIMESONE from dual;

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

**EXTRACT**

语法: EXTRACT(s)

功能:9i新增函数，从日期时间值中取得所需要的特定数据

Select extract(year from sysdate) year from dual;

Yaer

2013

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

**FROM\_TZ**

语法: FROM\_TZ(s)

功能:9i新增函数，将特定时区的TIMESTAMP值转换为TIMESTAMP WITH TIME ZONE值。

Select from\_tz(timestamp ‘2013-03-28 08:00:00’,’3:00’);

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

**LAST\_DAY**

语法:LAST\_DAY(d)

功能:计算包含日期的d的月份最后一天的日期.这个函数可以用来计算当月中剩余天数.

使用位置:过程性语言和SQL语句。

LAST\_DAY

返回日期的最后一天

SQL> select to\_char(sysdate,'yyyy.mm.dd') aa from dual;

aa

2004.05.09

SQL> select last\_day(sysdate) from dual;

LAST\_DAY(S

31-5月 -04

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

LOCALTIMESTAMP

语法：LOCALTIMESTAMP

功能：9i新增函数，返回当前会话时区的日期时间。

Select LOCALTIMESTAMP from dual;

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

**MONTHS\_BETWEEN**

语法:MONTHS\_BETWEEN(date1,date2)

功能:计算date1和date2之间相差的月数.如果date1<date2，则返回负数；如果date1,date2这两个日期中日分量信息是相同的,或者这两个日期都分别是所在月的最后一天,那么返回的结果是一个整数,否则包括一个小数,小数为富余天数除以31，Oracle以每月31天为准计算结果。

SQL> select months\_between('19-12月-1999','19-3月-1999') mon\_between from dual;

MON\_BETWEEN

  9

SQL>selectmonths\_between(to\_date('2000.05.20','yyyy.mm.dd'),to\_date('2005.05.20','yyyy.dd'))mon\_betw from dual;

MON\_BETW

-60

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

**NEW\_TIME**

语法:NEW\_TIME(d,zone1,zone2)

功能:计算当时区zone1中的日期和时间是d时候,返回时区zone2中的日期和时间.zone1和zone2是字符串. 给出在this时区=other时区的日期和时间。

使用位置:过程性语言和SQL语句。

NEW\_TIME   (d,   ‘tz1’,   ‘tz2’)

d:：一个有效的日期型变量

tz1   &   tz2:：下表中的任一时区

时区1   时区2      说明

AST      ADT           大西洋标准时间

BST      BDT           白令海标准时间

CST       CDT          中部标准时间

EST      EDT           东部标准时间

GMT                    格林尼治标准时间

HST      HDT           阿拉斯加—夏威夷标准时间

MST       MDT          山区标准时间

NST                    纽芬兰标准时间

PST      PDT           太平洋标准时间

YST      YDT           YUKON标准时间

SQL> select to\_char(sysdate,'yyyy.mm.dd hh24:mi:ss')bj\_time,to\_char(new\_time

2  (sysdate,'PDT','GMT'),'yyyy.mm.dd hh24:mi:ss') los\_angles fromdual;

BJ\_TIME            LOS\_ANGLES

2004.05.09 11:05:32 2004.05.09 18:05:32

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

**NEXT\_DAY**

语法:NEXT\_DAY(d,string)

功能: 给出日期d和星期string之后计算下一个星期的日期. String是星期几;当前会话的语言指定了一周中的某一天.返回值的时间分量与d的时间分量是相同的.String的内容可以忽略大小写.

使用位置:过程性语言和SQL语句。

NEXT\_DAY(date,'day')

SQL> select next\_day('18-5月-2001','星期五') next\_day from dual;

NEXT\_DAY

25-5月 -01

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

**NUMTODSINTERNAL**

语法：NUMTODSINTERNAL(n,char\_expr)

功能：将数字n转换为INTERNAL DAY TO SECOND格式， char\_expr可以是DAY\HOUR\MINUTE或SECOND。

Select NUMTODSINTERNAL(1000,’minute’) from dual;

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

**NUMTOYMINTERNAL**

语法：NUMTOYMINTERNAL(n,char\_expr)

功能：将数字n转换为INTERVAL YEAR TO MONTH格式，char\_expr可以是year或者month。

Select NUMTOYMINTERNAL(100,’MONTH’) from dual;

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

**ROUND**

语法:ROUND(d[,format])

功能:将日期d按照由format指定的格式进行四舍五入处理处理.如果没有给format则使用缺省设置`DD`.

使用位置:过程性语言和SQL语句。

Select round(sysdate,’MONTH’) from dual;

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

**SYS\_EXTRACT\_UTC**

语法：SYS\_EXTRACT\_UTC(date)

功能：返回特定时区时间所对应的格林威治时间。

Select SYS\_EXTRACT\_UTC(systimestamp) from dual;

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

**SYSDATE**

语法: SYSDATE

功能:取得当前的日期和时间,类型是DATE.它没有参数.但在分布式SQL语句中使用时,SYSDATE返回本地数据库的日期和时间.

使用位置:过程性语言和SQL语句。

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

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

**SYSTIMESTAMP**

语法：SYSTIMESTAMP

功能：9i新增函数，返回当前系统的日期时间及时区。

Select systimestamp from dual;

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

**TO\_DSINTERNAL**

语法：TO\_DSINTERNAL(char[,’nls\_param’])

功能：9i新增函数，将符合特定日期和时间格式的字符串转变为INTERVAL DAY TOSECOND类型。

Select TO\_DSINTERNAL(’58:10:10’) from dual;

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

**TO\_TIMESTAMP**

语法：TO\_TIMESTAMP(char[fmt[,’nls\_param’]])

功能：9i新增函数，将符合特定日期和时间格式的字符串转变为TIMESTAMP类型。

select systimestamp from dual

1. 字符型转成timestamp

Select TO\_TIMESTAMP(’01-1月-03’) fromdual;

select to\_timestamp('01-10月-0807.46.41.000000000 上午','dd-MON-yy hh:mi:ss.ffAM')

from dual;

2. timestamp转成date型

select cast(TO\_TIMESTAMP('2015-10-01 21:11:11.328', 'yyyy-mm-ddhh24:mi:ss.ff') as date)

from dual;

3. date型转成timestamp

select cast(sysdate as timestamp) date\_to\_timestamp

from dual;

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

**TO\_TIMESTAMP\_TZ**

语法：TO\_TIMESTAMP\_TZ(char[fmt[,’nls\_param’]])

功能：9i新增函数，将符合特定日期和时间格式的字符串转变为TIMESTAMP WITHTIME ZONE类型。

Select TO\_TIMESTAMP\_TZ(’20130101’,’yyyymmdd’) from dual;

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

**TO\_YMINTERNAL**

语法：TO\_YMINTERNAL(char)

功能：9i新增函数，将符合特定日期和时间格式的字符串转变为INTERVAL YEAR TOMONTH类型。

select TO\_TIMESTAMP('2015-10-01 21:11:11.328', 'yyyy-mm-dd hh24:mi:ss.ff')-

       TO\_TIMESTAMP('2015-10-0111:11:11.328', 'yyyy-mm-dd hh24:mi:ss.ff')

  from dual;

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

**TRUNC**

语法:TRUNC(d,format)

功能:截断日期时间数据，计算截尾到由format指定单位的日期d.缺省参数同ROUNG.

使用位置:过程性语言和SQL语句。如果fmt='mi'表示保留分,截断秒，如此类推。

SQL> select to\_char(trunc(sysdate,'hh'),'yyyy.mm.dd hh24:mi:ss') hh,

  2  to\_char(trunc(sysdate,'mi'),'yyyy.mm.dd hh24:mi:ss') hhmmfrom dual;

HH                 HHMM

2004.05.09 11:00:00 2004.05.09 11:17:00

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

**TZ\_OFFSET**

语法：TO\_OFFSET(time\_zone\_name||sessiontimezone||dbtimezone)

功能：9i新增函数，返回特定时区与UTC相比的时区偏移。

Select TO\_OFFSET (’EST’) from dual;

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

**F.5   转换函数**

(asciistr,bin\_to\_num,cast,chartorowid,compose,convert,decompose,hextoraw,INTERVAL,rawtonhex,rowidtochar,rowidtonchar,scn\_to\_timestamp,timestamp\_to\_scn,to\_char,to\_clob,to\_date,to\_lob,to\_label,to\_multi\_byte,to\_nchar,to\_number,to\_single\_byte,translate...using,unistr)

说明：用于将数值从一种数据类型转换为另一种数据类型。

**ASCIISTR**

语法：ASCIISTR(s)

功能：9i新增函数，将任意字符集的字符串转变为数据库字符集的ASCII字符串。

Select ASCIISTR (’中国’) 中 from dual;

中

\4E2D\56FD

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

**BIN\_TO\_NUM**

语法：BIN\_TO\_NUM(expr[,expr]…)

功能：9i新增函数，用于将位向量值转变为实际的数字值。

Select BIN\_TO\_NUM(1,0,1,1,1) 中from dual;

中

23

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

**CAST**

语法：CAST(expr AS type\_name)

功能：用于将一个内置数据类型或集合类型转变为另一个内置数据类型或集合类型。可以作用于长度为0的空字段视图建表格之用。

Select cast(SYSDATE AS VARCHAR2) 中from dual;

Create table tb\_dual nologging as Select cast(null as varchar2(1)) fromdual;

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

**CHARTOROWID**

语法:CHARTOROWID(string)

功能: 将字符数据类型转换为ROWID类型,把包含外部格式的ROWID的CHAR或VARCHAR2数值转换为内部的二进制格式.参数string必须是包含外部格式的ROWID的18字符的字符串.oracle7和oracle8中的外部格式是不同的.CHARTOROWID是ROWIDTOCHAR的反函数.

使用位置:过程性语言和SQL语句。

SQL> select rowid,rowidtochar(rowid),ename from scott.emp;

ROWID             ROWIDTOCHAR(ROWID) ENAME

AAAAfKAACAAAAEqAAA AAAAfKAACAAAAEqAAA SMITH

AAAAfKAACAAAAEqAAB AAAAfKAACAAAAEqAAB ALLEN

AAAAfKAACAAAAEqAAC AAAAfKAACAAAAEqAAC WARD

AAAAfKAACAAAAEqAAD AAAAfKAACAAAAEqAAD JONES

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

**COMPOSE**

语法：COMPOSE(string)

功能：9i新增函数，用于将输入字符串转变为UNICODE字符串值。

Select COMPOSE(‘o’||unistr(‘\0308’)) 中from dual;

中

?

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

**CONVERT**

语法:CONVERT(string,dest\_set[,source\_set])

功能:将字符串string从source\_set所表示的字符集转换为由dest\_set所表示的字符集.如果source\_set没有被指定,它缺省的被设置为数据库的字符集.

使用位置:过程性语言和SQL语句。

SQL> select convert('中国','US7ASCII','WE8ISO8859P1')"conversion" from dual;

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

**DECOMPOSE**

语法：DECOMPOSE(string)

功能：9i新增函数，用于分解字符串并返回相应的UNICODE字符串。

Select COMPOSE(‘chateoux’) 中from dual;

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

**HEXTORAW**

语法:HEXTORAW(string)

功能: 将string一个十六进制构成的字符串转换为二进制RAW数值. String中的每两个字符表示了结果RAW中的一个字节..HEXTORAW和RAWTOHEX为相反的两个函数.

使用位置:过程性语言和SQL语句。

Select HEXTORAW (‘AB56’) 中 from dual;

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

**INTERVAL**

语法：INTERVAL 'integer [- integer]' {YEAR | MONTH} [(precision)][TO {YEAR| MONTH}]

功能：该数据类型常用来表示一段时间差, 注意时间差只精确到年和月. precision为年或月的精确域, 有效范围是0到9, 默认值为2。

INTERVAL '123-2' YEAR(3) TO MONTH

表示: 123年2个月,"YEAR(3)" 表示年的精度为3, 可见"123"刚好为3为有效数值, 如果该处YEAR(n), n<3就会出错, 注意默认是2.

INTERVAL '11:12:10.1234567' HOUR TO SECOND

表示：小时，秒

结果：+00 11:12:10.123457

INTERVAL '123' YEAR(3)

表示: 123年0个月

INTERVAL '300' MONTH(3)

表示: 300个月, 注意该处MONTH的精度是3啊.

INTERVAL '4' YEAR

表示: 4年, 同 INTERVAL'4-0' YEAR TO MONTH 是一样的

INTERVAL '50' MONTH

表示: 50个月, 同 INTERVAL'4-2' YEAR TO MONTH 是一样

INTERVAL '123' YEAR

表示: 该处表示有错误, 123精度是3了, 但系统默认是2, 所以该处应该写成INTERVAL '123' YEAR(3) 或"3"改成大于3小于等于9的数值都可以的

INTERVAL '5-3' YEAR TO MONTH + INTERVAL '20' MONTH =

INTERVAL '6-11' YEAR TO MONTH

表示: 5年3个月 + 20个月 = 6年11个月

**RAWTONHEX**

语法:RAWTONHEX(rawvalue)

功能:9i新增函数，将RAW类数值rawvalue转换为一个相应的十六进制表示的字符串. rawvalue中的每个字节都被转换为一个双字节的字符串. RAWTOHEX和HEXTORAW是两个相反的函数.

使用位置:过程性语言和SQL语句。

Select rawtonhex(‘7D’) from dual;

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

**ROWIDTOCHAR**

语法:ROWIDTOCHAR(rowid)

功能:9i新增函数，将ROWID类型的数值rowid转换为varchar2的字符串表示,在oracle7和oracle8之间有些不一样的地方. ROWIDTOCHAR和CHARTOROWID是两个相反的函数.

使用位置:过程性语言和SQL语句。

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

**ROWIDTONCHAR**

语法:ROWIDTOCHAR(rowid)

功能:9i新增函数，将ROWID类型的数值rowid转换为Nvarchar2的字符串表示,在oracle7和oracle8之间有些不一样的地方. ROWIDTOCHAR和CHARTOROWID是两个相反的函数.

使用位置:过程性语言和SQL语句。

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

**SCN\_TO\_TIMESTAMP**

语法:SCN\_TO\_TIMESTAMP(number)

功能:10g新增函数，根据输入的scn值返回对应的大概日期时间，其中number用于指定scn值.

使用位置:过程性语言和SQL语句。

Select SCN\_TO\_TIMESTAMP(ora\_rowscn) from emp;

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

**TIMESTAMP\_TO\_SCN**

语法:TIMESTAMP\_TO\_SCN(timestamp)

功能:10g新增函数，用于根据输入的timestamp返回所对应的scn值，其中timestamp、用于指定日期时间。

使用位置:过程性语言和SQL语句。

Select TIMESTAMP\_TO\_SCN(order\_date) from emp;

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

**TO\_CHAR**

语法1:TO\_CHAR(character)

功能1:用于将NCHAR,NVARCHAR2,CLOB,NCLOB数据转变为数据库字符集数据，当用于NCHAR,NVARCHAR2,NCLOB时字符用单引号括起来，前面加上n。

Select to\_char（n’中国’） from dual;

语法2:TO\_CHAR(d [,format[,nlsparams]])

功能2:将日期d转换为一个VARCHAR2类型的字符串.format指定日期格式,.如果没有给定format,使用的就是该会话的缺省日期格式.nlsparams指定NLS参数. nlsparams的格式是:“NLS\_DATE\_LANGUAGE”

使用位置:过程性语言和SQL语句。

select to\_char(sysdate,'yyyy/mm/dd hh24:mi:ss') from dual;

2004/05/09 21:14:41

语法3:TO\_CHAR(labels[,format])

功能3:将MISLABEL的LABEL转换为一个VARCHAR2类型的变量.

使用位置:在trusted数据库的过程性语句和SQL语句。

语法4: TO\_CHAR(num[,format[,nlsparams]])

功能4:将NUMBER类型的参数num转换为一个VARCHAR2类型的变量.如果指定了format,那么它会控制这个转换处理.表5-5列除了可以使用的数字格式.如果没有指定format,它会控制这个转换过程.下面列出了可以使用的数字格式.如果没有指定format,那么结果字符串将包含和num中有效位的个数相同的字符. nlsparams用来指定小数点和千分位分隔符和货币符号.可以使用的格式:`NLS\_NUMERIC\_CHARS=”dg”NLS\_CURRENCY=”string”

d和g分别表示列小数点和千分位分隔符. String表示了货币的符号.例如,在美国小数点分隔符通常是一个句点(.),分组分隔符通常是一个逗号(,),而千分位符号通常是一个$.

使用位置:过程性语言和SQL语句。

SELECT TO\_CHAR(TO\_DATE('11-oct-2007'), 'fmDdthsp "of" Month,Year') FROM DUAL;

以上正确，需要注意的是不属于转换日期格式标识符需要使用双引号，如上面的"of"

SELECT promo\_name, TRIM(TO\_CHAR(promo\_end\_date,'Day')) ||', '||

                  TRIM(TO\_CHAR(promo\_end\_date,'Month')) ||' '||

                  TRIM(TO\_CHAR(promo\_end\_date,'DD, YYYY')) AS last\_day

       FROM promotions;

等价于下面，fm有trim的作用去掉多余空格

SELECT promo\_name,TO\_CHAR(promo\_end\_date,'fmDay')||','||

                 TO\_CHAR(promo\_end\_date,'fmMonth')||' '||

                 TO\_CHAR(promo\_end\_date,'fmDD, YYYY') AS last\_day

       FROM promotions;

上述当中，Day\DAY\day等等转换后都是带空格的，而YYYY则不会。

--将number格式转换为货币格式，前面均带空格

select TO\_CHAR(12345,'$99999D99') from dual;-- $12345.00

SELECT TO\_CHAR(1890.55,'$00G000D00') FROM DUAL;-- $01,890.55

SELECT TO\_CHAR(1890.55,'$99G999D99') FROM DUAL;--   $1,890.55

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

**TO\_CLOB**

语法:TO\_CLOB (char)

功能:9i新增函数，将字符串转变为CLOB类型。Char参数使用NCHAR,NVARCHAR2，NCLOB类型，字符串需要单引号括起来，且在前面加上n.

Select TO\_CLOB（n’中国’） from dual;

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

**TO\_DATE**

语法:TO\_DATE(String[,format[,nlsparams]])

功能:将符合特定日期格式的字符串转变为date类型. format是一个日期格式字符串.当不指定format的时候,使用该会话的缺省日期格式，需要特别注意的，缺省格式并不适用'2015-03-03'这种形式。

Select to\_date(‘20130101’,’yyyymmdd’) from dual;--正确

SELECT TO\_DATE('01/JANUARY/2007') FROM DUAL;--正确，缺省支持

SELECT TO\_DATE('01-JANUARY-2007') FROM DUAL;--正确，缺省支持

SELECT TO\_DATE('2015-03-03') FROM DUAL;--错误，缺省不支持

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

**TO\_LOB**

语法:TO\_LOB (long\_column)

功能:9i新增函数，将LONG或LONGROW列的数据转变为相应的LOB类型。但需要注意的是，在单纯的select语句中会报错，如例子所示。

使用位置:过程性语言和SQL语句。

例子：to\_lob转化long

select VIEW\_NAME,to\_lob(text) text from user\_views;  --会报错

create table temp\_liutao nologging as select VIEW\_NAME,to\_lob(text) textfrom user\_views  --通过

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

**TO\_LABEL**

语法:TO\_LABEL(String[,format])

功能:将String转换为一个MLSLABEL类型的变量. String可以是VARCHAR2或者CHAR类型的参数.如果指定了format,那么它就会被用在转换中.如果没有指定format,那么使用缺省的转换格式.

使用位置:过程性语言和SQL语句。

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

**TO\_MULTI\_BYTE**

语法:TO\_MULTI\_BYTE(String)

功能:计算所有单字节字符都替位换位等价的多字节字符的String.该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效.否则, String不会进行任何处理.TO\_MULTI\_BYTE和TO\_SINGLE\_BYTE是相反的两个函数.

使用位置:过程性语言和SQL语句。

SQL>  select to\_multi\_byte('高')from dual;

TO

高

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

**TO\_NCHAR**

语法1:TO\_NCHAR(char)

功能1:将字符串由数据库字符集转变为民族字符集。

SQL>  select TO\_NCHAR ('高')from dual;

语法2:TO\_NCHAR(date,[,fmt[,nls\_param]])

功能2:将日期时间值转变为民族字符集。

SQL>  select TO\_NCHAR (sysdate) from dual;

语法3:TO\_NCHAR(number)

功能3:将数字值转变为民族字符集。

SQL>  select TO\_NCHAR (10) from dual;

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

**TO\_NUMBER**

语法: TO\_NUMBER(String[,format[,nlsparams]])

功能:将CHAR或者VARCHAR2类型的String转换为一个NUMBER类型的数值.如果指定了format,那么String应该遵循相应的数字格式. Nlsparams的行为方式和TO\_CHAR中的完全相同.TO\_NUMBER和TO\_CHAR是两个相反的函数.

使用位置:过程性语言和SQL语句。

SQL> select to\_number('1999') year from dual;

     YEAR

     1999

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

**TO\_SINGLE\_BYTE**

语法: TO\_SINGLE\_BYTE(String )

功能:计算String中所有多字节字符都替换为等价的单字节字符.该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效.否则, String不会进行任何处理.

TO\_MULTI\_BYTE和TO\_SINGLE\_BYTE是相反的两个函数.

使用位置:过程性语言和SQL语句。

Select TO\_SINGLE\_BYTE(‘a b c’) from dual;

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

**TRANSLATE…USING**

语法: TRANSLATE(str1 USING zfj)

功能:将字符串转变为数据库字符集(char\_cs)或民族字符集(nchar\_cs)

Select TRANSLATE(‘中国’ usingnchar\_cs) from dual;

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

**UNISTR**

语法: UNISTR(str1)

功能:9i新增函数，输入字符串返回相应的UNICODE字符

Select UNISTR (‘\00D6’) from dual;

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

**F.6  分组统计函数**

(avg,corr,count,covar\_pop,covar\_samp,cume\_dist,dense\_rank,first,group\_id,grouping,grouping\_id,glb,last,listagg,lub,max,min,percent\_rank,percentile\_cont,percentile\_disc,rank,stddev,stddev\_pop,stddev\_samp,sum,var\_pop,var\_samp,variance)

说明：分组函数也被称为多行函数，它会根据输入的多行数据返回一个结果。主要用于执行数据统计或汇总操作，并且分组函数只能出现在select语句选择列表、order by子句和having子句中。注意分组函数不能直接在plsql中引用，只能在内嵌select语句中使用。

**AVG**

语法:AVG([DISTINCT|ALL]col)

功能:返回一列数据的平均值,缺省使用是ALL修饰符，all表示对所有的值求平均值,distinct排重后再求平均值

使用位置:查询列表和GROUP BY子句.

SQL> select avg(distinct sal) from gao.table3;

AVG(DISTINCTSAL)

         3333.33

SQL> select avg(all sal) from gao.table3;

AVG(ALLSAL)

    2592.59

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

**CORR**

语法:CORR([expr1,expr2)

功能:返回成对数值的相关系数，其数值使用表达式”covar\_pop(expr1,expr2)/(stddev\_pop(expr1)\*stddev\_pop(expr2))”

使用位置:查询列表和GROUP BY子句.

SQL> select corr(list\_,min\_) from gao.table3;

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

**COUNT**

语法:COUNT(\*|[DISTINCT|ALL] col)

功能:得到查询中行的数目.如果使用了\*获得行的总数.如果在参数中传递的是选择列表,那么计算的是非空数值。我基于10G测试，有主键情况下，count(主键)最快，count(1)和count(\*)调动主键统计，时间上一样；无主键情况下count(索引列)最快，但需要注意count(列名)统计不包括null，count(常量)和count(\*)包括null

Select count(distinct sal) from emp;

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

**COVAR\_POP**

语法:COVAR\_POP(expr1,expr2)

功能:返回成对数字的协方差，其数值使用表达式”(sum(expr1\*expr2)-sum(expr1)\*sum(expr2)/n)/n”

Select COVAR\_POP(column1,column2) from emp;

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

**COVAR\_SAMP**

语法:COVAR\_SAMP(expr1,expr2)

功能:返回成对数字的协方差，其数值使用表达式”(sum(expr1\*expr2)-sum(expr1)\*sum(expr2)/n)/n-1”

Select COVAR\_SAMP(column1,column2) from emp;

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

**CUME\_DIST**

语法:CUME\_DIST(expr1,expr2…) within group (order by expr1,expr2…)

功能:返回特定数值在一组行数据中的累积分布比例。

Select CUME\_DIST(2000) within group (order by sel) from emp;

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

**DENSE\_RANK**

语法:DENSE\_RANK(expr1,expr2…) within group (order by expr1,expr2…)

功能:返回特定数据在一组行数据中的等级。

Select DENSE\_RANK (5000) within group (order by sel) from emp;

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

**FIRST**

语法:FIRST

功能:9i新增，不能单独使用，必须与其他分组函数结合使用。通过使用该函数，可以取得排序等级的第一级，然后然后使用分组函数汇总该等级的数据。

Select min(sal) keep (dense\_rank first order by comm desc) from emp;

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

**GROUP\_ID**

语法:GROUP\_ID

功能:9i新增，用于区分分组结果中的重复行。

Select deptno,job,avg(sal),group\_id() from emp group bydeptno,rollup(deptno,job);

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

**GROUPING**

语法：GROUPING(expr)

功能:用于确定统计结果是否使用了特定的表达式，返回0则用到了表达式，1则未用。

例如:select corp\_code,org\_level,count(1),

       grouping(corp\_code),

       grouping(org\_level)

  from tb\_sys\_organization

group by rollup(corp\_code, org\_level);

select case grouping(corp\_code)

         when 1 then 'all\_corp'else corp\_code end corp\_code,

       case grouping(org\_level)

         when 1 then 'all\_org'else org\_level end org\_level,

       count(1)

  from tb\_sys\_organization

group by rollup(corp\_code, org\_level);

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

**GROUPING\_ID**

语法:GROUPING\_ID(expr1[,expr2]…)

功能:9i新增，用于返回对应于特定行的grouping位向量的值。

Select deptno,job,sum(sal),grouping\_id(job,deptno) from emp group byrollup(deptno,jon)

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

**GLB**

语法:GLB ([DISTINCT|ALL]label)

功能:获得由label界定的最大下界.函数仅用于trusted oracle.

使用位置:trusted数据库的选择列表和GROUP BY子句.

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

**LAST**

语法:LAST

功能:9i新增，不能单独使用，必须与其他分组函数结合使用。通过使用该函数，可以取得排序等级的最后一级，然后使用分组函数汇总该等级的数据。

Select min(sal) keep (dense\_rank last order by comm) from emp;

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

**LISTAGG**

语法:listagg

功能:列转行

select listagg(o.rybs, ';') within group(order by o.rybs)

  from gk\_xszrr o

 where rownum <= 100;

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

**LUB**

语法:LUB ([DISTINCT|ALL]label)

功能:获得由label界定的最小上界.用于trusted oracle.数据库.

使用位置:trusted数据库的选择列表和GROUP BY子句.过程性语言和SQL语句。

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

**MAX**

语法:MAX([DISTINCT|ALL]col)

功能:获得选择列表或表达式的最大值，ALL表示对所有的值求最大值,DISTINCT表示对不同的值求最大值,相同的只取一次

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select max(distinct sal) from scott.emp;

MAX(DISTINCTSAL)

            5000

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

**MIN**

语法:MIN([DISTINCT|ALL]col)

功能:获得选择列表或表达式的最小值，ALL表示对所有的值求最小值,DISTINCT表示对不同的值求最小值,相同的只取一次

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select min(all sal) from gao.table3;

MIN(ALLSAL)

    1111.11

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

**PERCENT\_RANK**

语法:PERCENT\_RANK(expr1,expr2…)WITHIN GROUP (ORDER BY expr1,expr2…)

功能:该函数用于返回特定数值在统计级别中所占的比例。

SQL> select percent\_rank(3000) within group(order by sal) from emp;

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

**PERCENTILE\_CONT**

语法:PERCENTILE\_CONT(percent\_expr)WITHIN GROUP (ORDER BY expr)

功能:9i新增，用于返回在统计级别中处于某个百分点的特定数值（按照连续分布模型确定）。

SQL> select percentile\_cont(.6) within group(order by sal) from emp;

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

**PERCENTILE\_DISC**

语法:PERCENTILE\_DISC(percent\_expr)WITHIN GROUP (ORDER BY expr)

功能:9i新增，用于返回在统计级别中处于某个百分点的特定数值（按照离散分布模型确定）。

SQL> select percentile\_cont(.6) within group(order by sal) from emp;

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

**RANK**

语法:RANK(expr1,expr2…)WITHIN GROUP (ORDER BY expr1,expr2…)

功能:该函数用于返回特定数值中所占据的等级。

SQL> select rank(3000) within group(order by sal) from emp;

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

**STDDEV**

语法:STDDEV([DISTINCT|ALL]col)

功能:获得选择列表的标准差.标准差为方差(VARIANCE)的平方根, ALL表示对所有的值求标准差,DISTINCT表示只对不同的值求标准差.

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select stddev(sal) from scott.emp;

STDDEV(SAL)

  1182.5032

SQL> select stddev(distinct sal) from scott.emp;

STDDEV(DISTINCTSAL)

           1229.951

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

**STDDEV\_POP**

语法:STDDEV\_POP(col)

功能:返回统计标准差，其数值是统计方差的平方根.

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select stddev\_pop(sal) from scott.emp;

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

**STDDEV\_SAMP**

语法:STDDEV\_SAMP(col)

功能:返回采样标准差，其数值是采样方差的平方根.

SQL> select stddev\_samp(sal) from scott.emp;

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

**SUM**

语法:SUM([DISTINCT|ALL]col)

功能:返回选择的数值和总和

使用位置:仅用于查询选择和GROUP BY子句.

Select sum(sal) from emp;

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

**VAR\_POP**

语法:VAR\_POP([DISTINCT|ALL]col)

功能:返回统计方差.使用公式为(sum(expr\*expr)-sum(expr)\*sum(expr)/count(expr))/(count(expr)

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select VAR\_POP (sal) from scott.emp;

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

**VAR\_SAMP**

语法:VAR\_SAMP([col)

功能:返回采样方差.使用公式为(sum(expr\*expr)-sum(expr)\*sum(expr)/count(expr))/(count(expr-1)

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select variance(sal) from scott.emp;

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

**VARIANCE**

语法:VARIANCE([DISTINCT|ALL]col)

功能:返回选择列或表达式的采样方差.使用公式为(sum(expr\*expr)-sum(expr)\*sum(expr)/count(expr))/(count(expr-1)

使用位置:仅用于查询选择和GROUP BY子句.

SQL> select variance(sal) from scott.emp;

VARIANCE(SAL)

    1398313.9

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

分组函数，除了count(\*)，count(1)，其他分组函数都会忽略null行，包括count(列名)。

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

**F.7  集合函数**

(cardinality,collect,powermultiset,powermultiset\_by\_cardinality,set)

说明：10g新增，为了扩展集合类型（嵌套表和VARRAY）的功能，新增的针对集合类型的函数。

**CARDINALITY**

语法:CARDINALITY (nested\_table)

功能:10g新增函数，返回嵌套表的实际元素个数。

SQL> select product\_id,CARDINALITY(ad\_text) from a;

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

**COLLECT**

语法:COLLECT (column)

功能:10g新增函数，用于根据输入列和被选中行建立嵌套表结果。

SQL> select cast(COLLECT(ad\_text) as t) from a;--t是嵌套表

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

**POWERMULTISET**

语法:POWERMULTISET(expr)

功能:10g新增函数，用于生成嵌套表的超集（包含所非空的嵌套表）。

SQL> select cast(POWERMULTISET (ad\_text) as t) from a;--t是嵌套表

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

**POWERMULTISET\_BY\_CARDINALITY**

语法:POWERMULTISET\_BY\_CARDINALITY(expr,cardinatility)

功能:10g新增函数，用于根据嵌套表和元素个数，生成嵌套表的超集（包含所非空的嵌套表）。

SQL> select cast(POWERMULTISET\_BY\_CARDINALITY(ad\_text) as t) from a;--t是嵌套表

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

**SET**

语法:SET(nested\_table)

功能:改函数用于取消嵌套表中的重复结果，并生成新的嵌套表。

SQL> select SET(nested\_table) from a;

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

**F.8  对象函数**

(deref,make\_ref,ref,reftohex,value)

说明：对象函数用于操纵REF对象。REF对象实际是指对象类型数据的指针。

**DEREF**

语法:DEREF(expr)

功能:该函数用于返回参照对象exp所引用的对象实例。

SQL> select DEREF(address).city from table\_name;

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

**MAKE\_REF**

语法:MAKE\_REF(object\_table|object\_view,key)

功能:该函数可以基于对象视图或对象表（存在基于主键的对象标识符）的一行数据建立REF。

SQL> select MAKE\_REF(oc\_inventocies,3003) from dual;

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

**REF**

语法:REF(expr)

功能:该函数用于返回对象行所对应的REF值。

SQL> select REF(e) from table\_name e;

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

**REFTOHEX**

语法:REFTOHEX(expr)

功能:该函数用于将REF值转变为十六进制字符串。

SQL> select REFTOHEX(REF(e)) from table\_name e;

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

**VALUE**

语法:VALUE(expr)

功能:该函数用于返回行对象所对应的对象实例数据，其中expr用于指定行对象的别名。

SQL> select value(e).city from table\_name e;

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

**F.9  其他函数**

(bfilename,coalesce,decode,depth,dump,empty\_clob/empty\_blob,existsnode,extract,extractvalue,greatest,greatest\_lb,least,least\_ub,nls\_charset\_decl\_len,nls\_charser\_id,nls\_charser\_name,nullif,nvl2,over,path,sys\_connect\_by\_path,sys\_context,sys\_dburigen,sys\_guid,sys\_typeid,sys\_xmlagg,sys\_xmlgen,uid,updatexml,user,userenv,vsize,xmlagg,xmlcolatival,xmlconcat,xmlelement,xmlforest,xmlsequence,xmltransform)

说明：除了上述涉及的函数外，Oracle还提供了一些单行函数。

**BFILENAME**

语法: BFILENAME(directory,file\_name)

功能:获得操作系统中与物理文件file\_name相关的BFILE位置指示符. Directory是与OS路径相关的DIRECTORY类型对象，file\_name是OS文件的名称。

使用位置:过程性语言和SQL语句。

SQL>insert into file\_tb1 values(bfilename('lob\_dir1','image1.gif'));

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

**COALESCE**

语法:COALESCE(exp1,exp2,exp3,...)

功能:9i新增，依次查找各参数，遇到非NULL则返回，各参数或表达式数据类型必须一致，如果都为null则返回null。

Select COALESCE(v\_e1,v\_e2) from a;

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

**DECODE**

语法:DECODE(base\_expr,comparel,valuel,Compare2,value2,…default)

功能:把base\_expr与后面的每个compare(n)进行比较,如果匹配返回相应的value (n).如果没有发生匹配,则返回default，每个valuel数据类型必须一致，如果没有default则返回null。

Select decode(a,’金’,1,’银’,2,0) from table\_name;

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

**DEPTH**

语法:DEPTH(n)

功能:9i新增，用于返回xml方案under\_path路径所对应的相对层数，其中参数n用于指定相对层数。

Select fath(1),depth(2) from a;

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

**DUMP**

语法:DUMP(expr[,number\_format[,start\_position][,length]])

功能:获得有关expr的内部表示信息的VARCHAR2类型的数值. number\_format指定了按照下面返回数值的基数(base):

number\_format    结果

8               八进制表示

10              十进制表示

16              十六进制表示

17              单字符

默认的值是十进制.

如果指定了start\_position和length,那么返回从start\_position开始的长为length的字节.缺省返回全部.

数据类型按照下面规定的内部数据类型的编码作为一个数字进行返回.

代码    数据类型

1       VARCHAR2

2       NUMBER

8       LONG

12      DATE

23      RAW

69      ROWID

96      CHAR

106     MLSLABEL

SQL> col global\_name for a30

SQL> col dump\_string for a50

SQL> set lin 200

SQL> select global\_name,dump(global\_name,1017,8,5) dump\_string fromglobal\_name;

GLOBAL\_NAME                   DUMP\_STRING

ORACLE.WORLD                  Typ=1 Len=12 CharacterSet=ZHS16GBK: W,O,R,L,D

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

**EMPTY\_CLOB/EMPTY\_BLOB**

语法:EMPTY\_CLOB()

     EMPTY\_BLOB()

功能:获得一个空的LOB提示符(locator).EMOTY\_CLOB返回一个字符指示符,而EMPTY\_BLOB返回一个二进制指示符, 用来对大数据类型字段进行初始化操作的函数.

使用位置:过程性语言和SQL语句.

Select EMPTY\_CLOB() from dual;

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

**EXISTSNODE**

语法:EXISTSNODE(XMLType\_instance,Xpatgh\_string)

功能:9i新增，用于确认xml节点路径是否存在，返回0表示不存在，1表示存在。.

使用位置:过程性语言和SQL语句.

Select EXISTSNODE(value(p),’/purchar/user’) from p;

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

**EXTRACT**

语法:EXTRACT (XMLType\_instance,Xpatgh\_string)

功能:9i新增，用于返回xml节点路径下的相应内容。.

使用位置:过程性语言和SQL语句.

Select EXTRACT (value(p),’/purchar/user’) from p;

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

**EXTRACTVALUE**

语法:EXTRACTVALUE(XMLType\_instance,Xpatgh\_string)

功能:9i新增，用于返回xml节点路径下的值。.

使用位置:过程性语言和SQL语句.

Select EXTRACTVALUE(value(p),’/purchar/user’) from p;

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

**GREATEST**

语法:GREATEST(expr1[,expr2]…)

功能:计算参数中最大的表达式.所有表达式的比较类型以expr1为准，比较字符的编码大小。

使用位置:过程性语言和SQL语句.

SQL> select greatest('AA','AB','AC') from dual;

GR

AC

SQL> select greatest('啊','安','天') from dual;

GR

安

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

**GREATEST\_LB**

语法:GREATEST\_LB(label1[,label2]…)

功能:返回标签(label)列表中最大的下界.每个标签必须拥有数据类型MLSLABEL、RAWMLSLABEL或者是一个表因字符串文字.函数只能用于truested oracle库.

使用位置:过程性语言和SQL语句.

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

**LEAST**

语法:LEAST(expr1[,expr2]…)

功能:计算参数中最小的表达式.所有表达式的比较类型以expr1为准，比较字符的编码大小。

使用位置:过程性语言和SQL语句.

SQL> select least('啊','安','天') from dual;

LE

啊

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

**LEAST\_UB**

语法:LEAST\_UB(label1[,label2]…)

功能:与GREATEST\_UB函数相似,本函数返回标签列表的最小上界.

使用位置:过程性语言和SQL语句.

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

**NLS\_CHARSET\_DECL\_LEN**

语法:NLS\_CHARSET\_DECL\_LEN(byte\_count,charset\_id)

功能:该函数用于返回字节数在特定字符集中占有的字符个数。

select NLS\_CHARSET\_DECL\_LEN(200,nls\_charset\_id(‘zhs16gbkf1xed’)) fromdual;

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

**NLS\_CHARSET\_ID**

语法:NLS\_CHARSET\_ID(text)

功能:该函数用于返回字符集的ID号。

select NLS\_CHARSET\_ID( ‘zhs16gbkf1xed’) from dual;

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

**NLS\_CHARSET\_NAME**

语法:NLS\_CHARSET\_NAME(number)

功能:该函数用于返回字符集ID号所对应的字符集名。

select NLS\_CHARSET\_NAME(852) from dual;

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

**NULLIF**

语法:NULLIF (expr1, expr2) ->相等返回NULL，不等返回expr1

功能：9i新增，用于比较表达式expr1和expr2，相等返回null，否则返回expr1.

Select nullif(expr1, expr2) from table\_name;

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

**NVL**

语法:NVL (expr1, expr2)

功能:用于将NULL转变为实际值，如果expr1是NULL,那么返回expr2,否则返回expr1，expr1、expr2两者必须为同类型或expr2可以隐式转换为expr1，否则会报错。

Select nvl(column\_name,0) from tbale\_name;

特别的date可以隐式转换为number，所以下面正确

SELECT NVL(to\_date('2017-01-01','yyyy-mm-dd')-sysdate,SYSDATE)

  FROM dual;

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

**NVL2**

语法:NVL2 (expr1, expr2, expr3)

功能:9i新增，expr1不为NULL，返回expr2；expr1为NULL，返回expr3。expr1可以是任意数据类型；expr2与expr3可以是除LONG外的任意数据类型，但需要类型一致或expr3可以隐式转换为expr2。

特别的date可以隐式转换为number，所以下面正确

SELECT NVL2(to\_date('01-jun-2016'),sysdate -to\_date('01-jun-2016'),sysdate)

  FROM dual;

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

**OVER**

语法：sun/count ( \* )

          over ( partition byXXX

      order by XXX )

功能：此函数为分析函数，有别于本文介绍中的其他函数，更详细看本博客“分析函数”专题

使用位置：过程性语言和SQL语句

sum(sal) over (partition by deptno order by ename)

COUNT( \* )

OVER (

PARTITION BY class\_id

ORDER BY ROWNUM

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

COUNT( \* )

OVER (PARTITION BY e.phone

ORDER BY pp.sort, qs.user\_id DESC, ROWNUM

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

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

**PATH**

语法:PATH(correction\_integer)

功能:9i新增，用于返回特定XML资源所对应的相对路径。

Select path(1),depth(2) from table\_name;

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

**SYS\_CONNECT\_BY\_PATH**

语法:SYS\_CONNECT\_BY\_PATH(column,char)

功能:9i新增（只适用于层次查询），用于返回从根到节点的列值路径。

Select lpad(‘ ‘,2\*level-1)||sys\_connect\_by\_path(ename,’/’) from table\_namestart with ename=’scott’ connect by prior empno=mgr;

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

**SYS\_CONTEXT**

语法：sys\_coniext(‘context’,’attribute’)

功能：该函数用于返回应用上下文的特定属性值，获得系统信息，其中context为上下文名，而attribute为应用上下文名，此函数可以得到oracle主机及客户端的信息。

SELECT   SYS\_CONTEXT ('USERENV', 'TERMINAL') 客户端名称,

         SYS\_CONTEXT ('USERENV','LANGUAGE') 客户端语言,

         SYS\_CONTEXT ('USERENV','SESSIONID') sessionid,

         SYS\_CONTEXT ('USERENV','INSTANCE') instance,

         SYS\_CONTEXT ('USERENV','ENTRYID') entryid,

         SYS\_CONTEXT ('USERENV','ISDBA') isdba,

         SYS\_CONTEXT ('USERENV','NLS\_TERRITORY') 地区,

         SYS\_CONTEXT ('USERENV','NLS\_CURRENCY') 货币,

         SYS\_CONTEXT ('USERENV','NLS\_CALENDAR') nls\_calendar,

         SYS\_CONTEXT ('USERENV','NLS\_DATE\_FORMAT') 时间格式,

         SYS\_CONTEXT ('USERENV','NLS\_DATE\_LANGUAGE') 时间语言,

         SYS\_CONTEXT ('USERENV','NLS\_SORT') nls\_sort,

         SYS\_CONTEXT ('USERENV','CURRENT\_USER') current\_user,

         SYS\_CONTEXT ('USERENV','CURRENT\_USERID') current\_userid,

         SYS\_CONTEXT ('USERENV','SESSION\_USER') session\_user,

         SYS\_CONTEXT ('USERENV','SESSION\_USERID') session\_userid,

         SYS\_CONTEXT ('USERENV','PROXY\_USER') proxy\_user,

         SYS\_CONTEXT ('USERENV','PROXY\_USERID') proxy\_userid,

         SYS\_CONTEXT ('USERENV','DB\_DOMAIN') db\_domain,

         SYS\_CONTEXT ('USERENV','DB\_NAME') 数据库名称,

         SYS\_CONTEXT ('USERENV','HOST') 客户端完成名称,

         SYS\_CONTEXT ('USERENV','OS\_USER') 客户端用户,

         SYS\_CONTEXT ('USERENV','EXTERNAL\_NAME') external\_name,

         SYS\_CONTEXT ('USERENV','IP\_ADDRESS') 客户端IP地址,

         SYS\_CONTEXT ('USERENV','NETWORK\_PROTOCOL') 网络协议,

         SYS\_CONTEXT ('USERENV','BG\_JOB\_ID') bg\_job\_id,

         SYS\_CONTEXT ('USERENV','FG\_JOB\_ID') fg\_job\_id,

         SYS\_CONTEXT ('USERENV','AUTHENTICATION\_TYPE') authentication\_type,

         SYS\_CONTEXT ('USERENV','AUTHENTICATION\_DATA') authentication\_data

  FROM   DUAL

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

**SYS\_DBURIGEN**

语法:SYS\_DBURIGEN(column)

功能:9i新增，根据列或属性生产类型为DBUriType的URL。

Select SYS\_DBURIGEN(ename) from emp;

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

**SYS\_GUID**

语法:SYS\_GUID()

功能:该函数用于生产类型为RAW的16字节的唯一标识符，每次调用该函数都会发生不同的RAW数据。

Select SYS\_GUID() from emp;

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

**SYS\_TYPEID**

语法:SYS\_TYPEID(object\_type\_value)

功能:该函数用于返回唯一的类型ID值。

Select name, SYS\_TYPEID(value(p)) from emp p;

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

**SYS\_XMLAGG**

语法:SYS\_XMLAGG(expr[,fmt])

功能:9i新增，用户汇总所有XML文档，并生成一个XML文档。

Select SYS\_XMLAGG(sys\_xmlgen(ename)) from emp p;

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

**SYS\_XMLGEN**

语法:SYS\_XMLGEN(expr[,fmt])

功能:9i新增，根据数据库表的行和列生成一个XMLType实例。

Select sys\_xmlgen(ename) from emp p;

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

**UID**

语法:UID

功能:获得当前数据库用的惟一标识,标识是一个整数.

使用位置:过程性语言和SQL语句.

SQL> show user

USER 为"GAO"

SQL> select username,user\_id from dba\_users where user\_id=uid;

USERNAME                        USER\_ID

GAO                                  25

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

**UPDATEXML**

语法:UPDATEXML(XMLType\_instance,Xpath\_string,value\_expr)

功能:9i新增，用于更新特定XMLType实例相对应节点路径的内容。

Update xmltable p set p=updatexml(value(p),’/pruch/user/text()’,’scott’)

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

**USER**

语法:USER

功能:取得当前oracle用户的名字,返回的结果是一个VARCHAR2型字符串.

使用位置:过程性语言和SQL语句.

SQL> select user from  dual;

USER

GAO

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

**USERENV**

语法:USERENV(option)

功能:根据参数option,取得一个有关当前会话信息的VARCHAR2数值.

使用位置:过程性语言和SQL语句.

返回当前用户环境的信息,opt可以是:

ENTRYID,SESSIONID,TERMINAL,ISDBA,LABLE,LANGUAGE,CLIENT\_INFO,LANG,VSIZE

OPTION='ISDBA'若当前是DBA角色,则为TRUE,否则FALSE.

OPTION='LANGUAGE'返回数据库的字符集.

OPTION='SESSIONID'为当前会话标识符.

OPTION='ENTRYID'返回可审计的会话标识符.

OPTION='LANG'返回会话语言名称的ISO简记.

OPTION='INSTANCE'返回当前的实例.

OPTION='terminal'返回当前计算机名

ISDBA  查看当前用户是否是DBA如果是则返回true

SQL> select userenv('isdba') from dual;

USEREN

FALSE

SESSION返回会话标志

SQL> select userenv('sessionid') from dual;

USERENV('SESSIONID')

                152

ENTRYID返回会话人口标志

SQL> select userenv('entryid') from dual;

USERENV('ENTRYID')

                0

INSTANCE返回当前INSTANCE的标志

SQL> select userenv('instance') from dual;

USERENV('INSTANCE')

                 1

LANGUAGE返回当前环境变量，包括语言、地区、字符集

SQL> select userenv('language') from dual;

USERENV('LANGUAGE')

SIMPLIFIED CHINESE\_CHINA.ZHS16GBK

LANG返回当前环境的语言的缩写

SQL> select userenv('lang') from dual;

USERENV('LANG')

ZHS

TERMINAL返回用户的终端或机器的OS标示符

SQL> select userenv('terminal') from dual;

USERENV('TERMINA

GAO

CLIENT\_INFO 返回由包DBMS\_APPLICATION\_INFO所存储的用户会话信息（64字节）

Select userenv(‘CLIENT\_INFO’) from dual;

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

**VSIZE**

语法:VSIZE(value)

功能:获得value的内部表示的字节数.如果value是NULL,结果是NULL.

使用位置: SQL语句.

SQL> select vsize(user),user from dual;

VSIZE(USER) USER

          6 SYSTEM

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

**XMLAGG**

语法:XMLAGG(XMLType\_instance [order by sort\_list])

功能:9i新增，用于汇总多个XML块，并生成XML文档。

Select xmlagg(xmlelement(“employee”,ename||’ ’||sal)) from emp wheredeptno=10;

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

**XMLCOLATTVAL**

语法:XMLCOLATTVAL(value\_expr1[,value\_expr2]…)

功能:9i新增，用于生成XML块，并增加”column”作为属性名。

Select xmlelement(“emp”, XMLCOLATTVAL(ename,sall)) from emp wheredeptno=10;

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

**XMLCONCAT**

语法:XMLCONCAT(XMLType\_instance1[,XMLType\_instance2]…)

功能:9i新增，用于连接多个XMLType实例，并生成一个新的XMLType实例。

Select XMLCONCAT(xmlelement(‘ename’,ename), xmlelement(‘sal’,sal)) fromemp where deptno=10;

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

**XMLELEMENT**

语法:XMLELEMENT(identifier[,xml\_attribute\_clause][,value\_expr])

功能:9i新增，用于返回XMLType实例，其中参数identifier指定元素名，参数xml\_attribute\_clause指定元素属性子句，参数value\_expr指定元素值。

Select XMLELEMENT(‘date’,sysdate) from dual;

Select XMLELEMENT(“emp”,xmlattributes(empno as “id”,ename)) from emp;

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

**XMLFOREST**

语法:XMLPOREST(value\_expr1[,value\_expr2]…)

功能:9i新增，用于返回XML块。

Select xmlelement(‘ename’, XMLPOREST[ename,sal]) from emp where deptno=10;

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

**XMLSEQUENCE**

语法:XMLSEQUENCE(xmltype\_instance)

功能:9i新增，用于返回xmltype实例中顶级节点一下的varray元素。

Select XMLSEQUENCE(extract(value(x),’/purorder/line/\*’)) from emp p wheredeptno=10;

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

**XMLTRANSFORM**

语法:XMLTRANSFORM(xmltype\_instance,xsl\_ss)

功能:9i新增，用于将xmltype实例按照XSL样式进行转换，并生成新的xmltype实例。

Select XMLTRANSFORM(w.warehouse\_spec,x.coll).getclobval from warehousew,xsl\_tab x where w.name=x.name;

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

**十、Oracle备份与恢复**

**Oracle备份方式分类：**  
Oracle有两类备份方式：  
（1）物理备份：是将实际组成数据库的操作系统文件从一处拷贝到另一处的备份过程，通常是从磁盘到磁带。  
物理备份又分为冷备份、热备份；    
（2）逻辑备份：是利用SQL语言从数据库中抽取数据并存于二进制文件的过程。  
逻辑备份使用导入导出工具：EXPDP/IMPDP或EXP/IMP；

备份方式的优缺点及使用时机比较如下图：

**Oracle备份方式介绍：  
1、物理备份之冷备份(条件－NonArchiveLog)：**  
　　当数据库可以暂时处于关闭状态时，我们需要将它在这一稳定时刻的数据相关文件转移到安全的区域，当数据库遭到破坏，再从安全区域将备份的数据库相关文件拷贝回原来的位置，这样，就完成了一次快捷安全等数据转移。由于是在数据库不提供服务的关闭状态，所以称为冷备份。冷备份具有很多优良特性，比如上面图中我们提到的，快速，方便，以及高效。一次完整的冷备份步骤应该是：  
（1）首先关闭数据库（shutdown normal）  
（2）拷贝相关文件到安全区域（利用操作系统命令拷贝数据库的所有的数据文件、日志文件、控制文件、参数文件、口令文件等（包括路径））  
（3）重新启动数据库（startup）  
以上的步骤我们可以用一个脚本来完成操作：  
　　su – oracle<      sqlplus /nolog   
　　connect / as sysdba  
　　shutdown immediate;  
　　!cp 文件   备份位置（所有的日志、数据、控制及参数文件）;  
　　startup;  
　　exit;  
这样，我们就完成了一次冷备份，请确定你对这些相应的目录（包括写入的目标文件夹）有相应的权限。  
物理冷备份的恢复：  
恢复的时候，相对比较简单了，我们停掉数据库，将文件拷贝回相应位置，重启数据库就可以了，当然也可以用脚本来完成。

**2、物理备份之热备份：(条件－ArchiveLog)**  
　　当我们需要做一个精度比较高的备份，而且我们的数据库不可能停掉（少许访问量）时，这个情况下，我们就需要归档方式下的备份，就是下面讨论的热备份。热备份可以非常精确的备份表空间级和用户级的数据，由于它是根据归档日志的时间轴来备份恢复的，理论上可以恢复到前一个操作，甚至就是前一秒的操作。具体步骤如下：  
（1）通过视图v$database,查看数据库是否在Archive模式下:          
    SQL> select log\_mode from v$database;  
　　如果不是Archive模式，则设定数据库运行于归档模式下：  
   SQL>shutdown immediate  
　　SQL>startup mount  
　　SQL>alter databasearchivelog;  
　　SQL>alter databaseopen;  
　　如果Automaticarchival显示为“Enabled”，则数据库归档方式为自动归档。否则需要手工归档，或者将归档方式修改为自动归档，如：  
　　正常shutdown数据库，在参数文件中init.ora中加入如下参数  
　　SQL>shutdownimmediate  
　　 修改init.ora：  
　　LOG\_ARCHIVE\_START=TRUE  
　　LOG\_ARCHIVE\_DEST1=ORACLE\_HOME/admin/o816/arch(归档日志存放位置可以自己定义)  
　　SQL>startup  
　　然后，重新启动数据库，此时Oracle数据库将以自动归档的方式工作在Archive模式下。其中参数LOG\_ARCHIVE\_DEST1是指定的归档日志文件的路径，建议与Oracle数据库文件存在不同的硬盘，一方面减少磁盘I/O竞争，另外一方面也可以避免数据库文件所在硬盘毁坏之后的文件丢失。归档路径也可以直接指定为磁带等其它物理存储设备，但可能要考虑读写速度、可写条件和性能等因素。        
　　注意：当数据库处在ARCHIVE模式下时，一定要保证指定的归档路径可写，否则数据库就会挂起，直到能够归档所有归档信息后才可以使用。另外，为创建一个有效的备份,当数据库在创建时,必须履行一个全数据库的冷备份,就是说数据库需要运行在归档方式,然后正常关闭数据库,备份所有的数据库组成文件。这一备份是整个备份的基础,因为该备份提供了一个所有数据库文件的拷贝。（体现了冷备份与热备份的合作关系，以及强大的能力）  
（2）表空间文件备份步骤：  
　　a,首先，修改表空间文件为备份模式 ALTER TABLESPACE tablespace\_name BEGIN BACKUP;  
　　b,然后，拷贝表空间文件到安全区域 !CP tablespace\_name D\_PATH;  
　　c,最后，将表空间的备份模式关闭    ALTER TABLESPACE tablespace\_name END BACKUP;  
（3）归档日志文件备份步骤：  
　　停止归档进程－－>备份归档日志文件－－>启动归档进程  
　　如果日志文档比较多，我们将它们写入一个文件成为一个恢复的参考：$   files `ls <归档文件路径>/arch\*.dbf`;exportfiles  
（4）控制文件备份步骤:  
　　SQL> alter databasebackup controlfile to 'controlfile\_back\_name（一般用2004-11-20的方式）' reuse;  
　　当然，我们也可以将上面的东东写为一个脚本，在需要的时候执行就可以了:  
　　脚本范例：  
　　su – oracle<      sqlplus /nolog   
　　connect / as sysdba  
　　ALTER TABLESPACEtablespace\_name BEGIN BACKUP  
　　!CP tablespace\_nameD\_PATH  
　　ALTER TABLESPACEtablespace\_name END BACKUP  
　　alter database backupcontrolfile to 'controlfile\_back\_name（一般用2004-11-20的方式）' reuse;  
　　!files `ls <归档文件路径>/arch\*.dbf`;export files  
物理热备份的恢复： 　   
热备份的恢复，对于归档方式数据库的恢复要求不但有有效的日志备份还要求有一个在归档方式下作的有效的全库备份。归档备份在理论上可以无数据丢失，但是对于硬件以及操作人员的要求都比较高。在我们使用归档方式备份的时候，全库物理备份也是非常重要的。归档方式下数据库的恢复要求从全备份到失败点所有的日志都要完好无缺。  
　　恢复步骤：LOG\_ARCHIVE\_DEST\_1  
　　（1）shutdown数据库。  
　　（2）将全备份的数据文件放到原来系统的目录中。  
　　（3）将全备份到失败点的所有归档日志放到参数LOG\_ARCHIVE\_DEST\_1所指定的位置。  
　　（4）利用sqlplus登陆到空实例。（connect / as sysdba）  
　　       startup mount  
　　       set autorecovery on  
　　       recover database;  
　　       alter database open;

**3、逻辑备份之EXP/IMP：**  
EXP和IMP是客户端工具程序，它们既可以在客户端使用，也可以在服务端使用。  
EXPDP和IMPDP是服务端的工具程序，他们只能在ORACLE服务端使用，不能在客户端使用。  
IMP只适用于EXP导出的文件，不适用于EXPDP导出文件；IMPDP只适用于EXPDP导出的文件，而不适用于EXP导出文件。  
使用EXP/IMP导出/导入包括三种方式：  
　　（1）表方式(T)    可以将指定的表导出备份；  
    （2）用户方式(U)    可以将指定的用户相应的所有数据对象导出；  
　　（3）全库方式(Full)     将数据库中的所有对象导出；  
（1）导出表：  
E:\>exp system/oracleTABLES=scott.dept,scott.emp FILE=a.dmp  
（2）导出方案：  
E:\>exp system/oracleOWNER=scott FILE=b.dmp  
（3）导出数据库：  
E:\>exp system/oracleFILE=c.dmp FULL=Y  
（4）导入表：  
SQL> drop table scott.emp;  
SQL> drop table scott.dept;  
E:\>impdp scott/tigerfile=a.dmp tables=dept,emp  
（5）导入方案：  
SQL> drop user scottcascade;  
SQL> create user scottidentified by tiger;  
SQL> grant dba to scott;  
E:\>impdp scott/tigerfile=b.dmp  
E:\>impdp system/oraclefile=b.dmp owner=scott  
（6）导入数据库：  
impdp system/oracle file=c.dmpfull=y  
在导入导出备份方式中，提供了很强大的一种方法，就是增量导出/导入，但是它必须作为System来完成增量的导入导出，而且只能是对整个数据库进行实施。增量导出又可以分为三种类别：  
（1）完全增量导出(Complete Export)     这种方式将把整个数据库文件导出备份；exp system/manager inctype=complete file=20041125.dmp（为了方便检索和事后的查询，通常我们将备份文件以日期或者其他有明确含义的字符命名）  
（2）增量型增量导出(Incremental Export)     这种方式将只会备份上一次备份后改变的结果；exp system/manager inctype=incremental file=20041125.dmp  
（3）累积型增量导出(Cumulate Export)      这种方式的话，是导出自上次完全增量导出后数据库变化的信息。exp system/manager inctype=cumulative file=20041125.dmp  
通常情况下，DBA们所要做的，就是按照企业指定或者是自己习惯的标准(如果是自己指定的标准，建议写好计划说明)，一般，我们采用普遍认可的下面的方式进行每天的增量备份：  
　　 Mon： 完全备份（A）   
　　 Tue： 增量导出（B）   
　　 Wed： 增量导出（C）   
　　 Thu： 增量导出（D）   
　　 Fri： 累计导出（E）   
　　 Sat： 增量导出（F）   
　　 Sun： 增量导出（G）  
这样，我们可以保证每周数据的完整性，以及恢复时的快捷和最大限度的数据损失。恢复的时候，假设事故发生在周末，DBA可按这样的步骤来恢复数据库：   
　　 第一步：用命令CREATEDATABASE重新生成数据库结构；   
　　 第二步：创建一个足够大的附加回滚。   
　　 第三步：完全增量导入A：   
　　 imp system/managerinctype=RESTORE FULL=y FILE=A   
　　 第四步：累计增量导入E：   
　　 imp system/manager inctype=RESTOREFULL=Y FILE=E   
　　 第五步：最近增量导入F：   
　　 imp system/managerinctype=RESTORE FULL=Y FILE=F   
 通常情况下，DBA所要做的导入导出备份就算完成，只要科学的按照规律作出备份，就可以将数据的损失降低到最小，提供更可靠的服务。另外，DBA最好对每次的备份做一个比较详细的说明文档，使得数据库的恢复更加可靠。

**4、逻辑备份之EXPDP/IMPDP**  
使用数据泵（Data Dump）导出/导入包括4种方式：导出表、导出方案、导出表空间、导出数据库。  
（1）导出表  
在E盘建立目录oracledump；  
SQL> CREATE DIRECTORYdump\_dir AS 'E:\oracledump';  
SQL> GRANT READ,WRITE ONDIRECTORY dump\_dir TO scott;  
E:\>expdp scott/tigerDIRECTORY=dump\_dir DUMPFILE=tab.dmp TABLES=dept,emp  
（2）导出方案：  
E:\>expdp scott/tiger DIRECTORY=dump\_dirDUMPFILE=schemaScott.dmp SCHEMAS='SCOTT';  
（3）导出表空间：  
E:\>expdp system/oracledirectory=dump\_dir dumpfile=tablespaceUsers.dmp ESTIMATE\_ONLY  
（4）导出数据库：  
E:\>expdp system/oracledirectory=dump\_dir dumpfile=database.dmp FULL=Y  
（5）导入表：  
SQL> drop table scott.emp;  
SQL> drop table scott.dept;  
E:\>impdp scott/tigerdirectory=dump\_dir dumpfile=tab.dmp tables=dept,emp  
（6）导入方案：  
SQL> drop user scottcascade;  
SQL> create user scottidentified by tiger;  
SQL> grant dba to scott;  
E:\>impdp system/oracledirectory=dump\_dir dumpfile=schemaScott.dmp schemas=scott  
（7）导入表空间：  
impdp system/oracledirectory=dump\_dir dumpfile=tablespaceUsers.dmp tablespaces=users  
（8）导入数据库：  
impdp system/oracledirectory=dump\_dir dumpfile=database.dmp FULL=Y

**十一、Oracle故障与恢复**

非归档模式的**数据库**，丢失数据文件

l  **故障现象**

Ø  丢失某个数据库文件，造成了数据库无法启动，同时数据库处于非归档模式，也没有冷**备份**，启动时的错误信息如下：

ORA-01157: cannot identify/lock **data** file 3- see DBWR trace file

ORA-01110: data file 3:'D:\ORACLE\ORADATA\TEST\USERS01.DBF'

l  **解决方法**

Ø  将数据库启动到mount状态下：

Ø[SQL](http://www.dataguru.cn/article-8711-1.html?union_site=innerlink)plus “/ assysdba”

Østartup mount

Ø从数据库中删除该数据文件

Øalter **database** datafile ‘xx’ offline drop;

Ø  打开数据库

Øalter database open;

Ø备注:

Ø该方法可正常打开数据库，但该datafile中的数据将丢失

Ø如果误删除了system表空间的datafile,则该方法不奏效

Ø如果该表空间还包含其它数据文件，用EXP把数据备份出来，然后删除表空间，重建表空间，将数据导入。如果不包含其它数据文件，则直接删除表空间就可以了。

归档模式数据库丢失某数据文件，无备份,但有该数据文件创建以来的归档日志

l  **故障现象**

Ø归档模式的数据库，丢失了某个数据库文件，造成了数据库无法启动，同时没有数据库的全备份，但有该数据文件创建以来的归档日志，数据库无法启动：

       ORA-01157:cannot identify/lock data file 3 - see DBWR trace file

       ORA-01110:data file 3: 'D:\ORACLE\ORADATA\TEST\USERS01.DBF

l**解决方法**

Ø启动数据库到mount状态

Østartup mount

Ø手工创建丢失的数据文件

Øalter database create datafile ‘oldfname’ as  ‘newfname’size xxx reuse;

Ø  利用归档日志对数据文件进行**恢复**

Ø  recover datafile ‘newfname’;或者

Ørecover datafile  n;

Ø打开数据库

Øalter database open;

Ø备注:

Ø该方法可正常打开数据库，而且不会丢失数据

Ø该方法有两个前提

Ø  丢失的数据文件不能是系统文件

Ø  不能丢失或损坏控制文件

**非current和active的redo log损坏**

l  **故障现象**

Ø  误删除了redo log，或者redo log被损坏，数据库能mount,不能open:

       ORA-00313:open failed **for** members of log group 3 of thread 1

       ORA-00312:online log 3 thread 1: '/oracle10/oradata/ora10g/redo03.log'

l**解决方法**

Ø  查询v$log视图，确认损坏的redo log group是非current和active

ØSQL>select group#,thread#,sequence#, archived,status from v$log;

       GROUP# THREAD#  SEQUENCE#  ARCHIVED STATUS

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

       1               1               103                 YES      INACTIVE

       2               1               104                 NO       CURRENT

3               1               102                 YES      INACTIVE

Ø  如果该日志已经归档，用下面的命令清除日志内容

**Ø**Alter database clear logfile group 3;

**Ø**如果该日志没有归档，用下面的命令清除日志内容

**Ø**Alter database clear unarchived logfile group 3;

**Ø**打开数据库

**Ø**Alter database open;

**Ø**尽快做一个数据库全备份

**current或active的redo log损坏**

l  **故障现象**

**Ø**误删除了redo log，或者redo log被损坏，数据库不能打开:

       ORA-00313:open failed for members of log group 2 of thread 1

ORA-00312: online log 2 thread 1:'/oracle10/oradata/ora10g/redo02.log'

**l解决方法**

**Ø**查询v$log视图，确认损坏的redo log group是current或active

**Ø**SQL>select group#,thread#,sequence#, archived,statusfrom v$log;

       GROUP# THREAD#  SEQUENCE#  ARCHIVED STATUS

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

1               1                   2                     YES      INACTIVE

2               1                   4                     NO       CURRENT

3               1                   3                     YES      INACTIVE

**Ø**情况1：当前日志文件还存在，只是逻辑损坏，并且当前日志没有未决事务需要实例恢复

**Ø**alter database clear unarchived logfile group 2;  --不会报错

**Ø**recover database until cancel;

**Ø**alter database open resetlogs;

**Ø**一般情况下，该方法不奏效，如果clear报错，则用其它方法.

**Ø**情况2：当前日志完全损坏，且有未决事务，数据库有备份

**Ø**alter database clear unarchived logfile group 2;  --会报错

ERROR at line 1:

ORA-01624: log 1 needed for crash recovery of thread 1

**Ø**restore database;

**Ø**recover database until cancel; --选择auto

**Ø**recover database until cancel;

**Ø**alter datbase open resetlogs;

**Ø**尽快做一个数据库全备份

**Ø**情况3：当前日志完全损坏，且有未决事务，数据库无备份

**Ø**shutdown immediate;

**Ø**\_allow\_resetlogs\_corruption=true;

**Ø**startup mount pfile=‘xxx’;

**Ø**recover database until cancel;

**Ø**alter datbase open resetlogs;

**Ø**shutdown immediate

**Ø**\_allow\_resetlogs\_corruption=true;

**Ø**Startup

**Ø**尽快做一个数据库全备份

临时表空间的数据文件损坏

l  **故障现象**

Ø  临时表空间的数据文件发生损坏，系统出现故障，如何恢复

l**解决方法**

Ø  在**10g**及以上版本数据库，启动数据库时，如果发现临时数据文件损坏，会自动创建，如果在数据库运行过程中，可以手工重建:

Øcreate temporary tablespace temp1 tempfile ‘xx’ size xx’;

Øalter database default temporary tablespace temp1;--系统默认临时表空间的重建需要执行这一步，否则不需要

Ødrop tablespace temp;

Øalter tablespace temp1 rename to temp;

Ø  在10g以前版本数据库，可以在数据库打开后或运行过程中，手工重建就可以了

Ø  alter database datafile ‘xxx’ offline drop;--如果数据库打不开，就执行这个步骤

**Ø**create temporary tablespace temp1 tempfile ‘xx’ size xx’;

**Ø**alter database default temporary tablespace temp1;--系统默认临时表空间的重建需要执行这一步 ,否则不需要,9i以前版本也不需要。

**Ø**drop tablespace temp;

**Ø**alter tablespace temp1 rename to temp;

**UNDO数据文件损坏，数据库无法启动**

l  故障现象

Ø  Undo数据文件发生了丢失或损坏，数据库启动报错:

       ORA-01157:cannot identify/lock data file 2 - see DBWR trace file

       ORA-01110:data file 2: '/oracle10/oradata/ora10g/undotbs01.dbf'

l解决方法

Ø  如果数据库有备份，则利用备份进行恢复

Ø  如果数据库没有备份，则利用重建undo表空间的方式进行恢复

Østartup mount

Øalter database datafile n offline drop;(删除损坏的undo文件)

Øalter database open;

Øcreate undo tablespace xxx …; (创建一个新的undo表空间)

Øalter system set undo\_tablespace=xxx;(指向新的undo表空间)

Ødrop tablespace yyy including contents;(删除原来的undo表空间)

控制文件损坏

l  **故障现象**

Ø  控制文件发生了损坏，数据库已经无法启动，报错信息如下：

       ORA-00202:controlfile: 'D:\Oracle\oradata\chen\control01.ctl'

       ORA-27041:unable to open file

       OSD-04002:unable to open file

l**解决方法**

Ø  情况一：控制文件有镜像，且镜像控制文件没有被损坏

Ø关闭数据库

Ø将没有损坏的控制文件覆盖掉损坏的控制文件，或者修改参数文件的control\_files参数，去掉损坏的控制文件

Ø重新启动数据库

Ø  情况二：控制文件无镜像，或者镜像的所有控制文件都损坏了

Ø恢复控制文件

Ø如果控制文件有备份，从备份中恢复控制文件

       restorecontrolfile from ‘<your controlfilebackupset>’

Ø如果控制文件有snapshot,将snapshot控制文件替换掉原损坏控制文件

Ø如果做过alter database backup controlfile to trace的控制文件脚本备份，可以用trace文件中的重建脚本来创建控制文件，

Ø如果没有备份，也没有trace备份，只能手工编写脚本创建控制文件，前提是你对数据库文件结构非常清楚

Ø  恢复和打开数据库

Ø如果是用create controlfile …noresetlogs方式重建的控制文件

Ørecover database;

Øalter database open;

Øalter tablespace temp add tempfile ‘xx’ size xx reuse ; --对所有临时表空间做此操作

Ø如果是用create controlfile …resetlogs方式重建的控制文件，或者通过备份或快照恢复的控制文件

Ørecover database using backup controlfile;

Øalter database open resetlogs;

Øalter tablespace temp add tempfile ‘xx’ size xx reuse ; --对所有临时表空间做此操作

**当Oracle有多个实例时，执行conn /as  
sysdba出错**

使用以下命令设置默认实例

set  
oracl\_sid=XXX（XXX为希望设为默认的实例SID）

注：该命令在命令行下执行，并非sqlplus中。

**dba的密码遗忘，无法以dba账号（sys或system）登陆**

1）以本地用户身份进入sqlplus（前提：确保本地系统用户在oracle的DBA group中）

sqlplus /nolog

2）以dba身份登录

conn /as sysdba

3）修改dba sys或system的密码（新密码为123456）

alter user sys identified by  
123456;

**Oracle用户密码过期**

使用sqlplus登陆oracle数据库时提示“ORA-28002: 7 天之后口令将过期” 或提示 密码过期。

【原因/触发因素】  
由于oracle11g中默认在default概要文件中设置“PASSWORD\_LIFE\_TIME=180天”所导致。

处理方法及步骤：  
1、查看用户的proifle是哪个，一般是default：  
  
sql>SELECT username,PROFILE FROM dba\_users;  
  
   
  
2、查看指定概要文件(如default)的密码有效期设置：  
  
sql>SELECT \* FROM dba\_profiles s WHERE s.profile='DEFAULT' AND  
resource\_name='PASSWORD\_LIFE\_TIME';  
  
   
  
3、将密码有效期由默认的180天修改成“无限制”：  
  
sql>ALTER PROFILE DEFAULT LIMIT PASSWORD\_LIFE\_TIME  
UNLIMITED;  
  
修改之后不需要重启动数据库，会立即生效。  
  
   
  
4、修改后，还没有被提示ORA-28002警告的帐户不会再碰到同样的提示;  
  
已经被提示的帐户必须再改一次密码，举例如下：  
  
$sqlplus / as sysdba  
  
sql> alter user smsc identified by <原来的密码>  
----不用换新密码  
  
oracle11g启动参数resource\_limit无论设置为false还是true，密码有效期都是生效的，所以必须通过以上方式进行修改。以上的帐户名请根据实际使用的帐户名更改。

**Oracle数据库无法用IP连接**

Oracle数据库（Oracle  
11g）安装后，在本机使用sqldeveloper进行连接，如果主机名输入localhost，可以成功连接；但如果主机名输入IP地址，无法连接，提示信息为：Oracle  
the network adapter could not establish the connection。  
  
如果在命令行中输入 sqlplus [username/password@IP](mailto:username/password@IP)地址，系统提示“无监听程序”。

经百度，解决方案如下：  
1、开始-->程序-->Oracle -  
OraDb11g\_home1-->配置和移植工具-->Net Manager  
  
2、点击 监听程序 LISTENER，在 监听位置 页面中，增加地址：  
  
       
协议：TCP/IP  
  
       
主机：本地IP地址  
  
       
端口：1521  
  
3、点击菜单 文件-保存网络配置  
  
4、重启监听服务：可使用命令进行重启 lsnrctl reload；也可在服务中重启 TNSListener 服务

**安装64位版Oracle11gR2后无法启动SQLDeveloper**

安装64位版Oracle11gR2后发现启动SQL  
Developer时弹出配置java.exe的路径，找到Oracle自带java.exe后产生的路径“\app\用户名\product\11.2.0\dbhome\_1\jdk”却弹出错误信息：

－－－－－－－－－－－－－－－－－－－－－－－－－－  
  
Unable to find a java Virtual Machine  
  
to point to a location of a java virtual machine,please refer to  
the oracle9i Jdeveloper Install guide(jdev\install.html)  
  
－－－－－－－－－－－－－－－－－－－－－－－－－－

在安装目录下（app\用户名\product\11.2.0\dbhome\_1\sqldeveloper0\sqldeveloper\bin）手动修改配置文件sqldeveloper.conf中的“SetJavaHome”为电脑上独立安装的JDK，结果还是一样。

经百度，原来Oracle在制造64位版的时候没注意Oracle11gR2所带的SQL  
Developer是1.5.5.59.69版，不支持64位版的JDK，恰好64位Oracle带的JDK和“C:\Program  
Files”中的JDK都是64位的。如果你单独安装的JDK中“C:\Program Files  
(x86)”中则说明是32位版的，是可以用的。为什么MyEclipse所带的JDK可用呢，因为MyEcipse8.5没有64位版（包括最新的8.6也一样），所以其中带的JDK当然是32位版的了。

解决方案如下：

1）单独安装一个32位版的JDK就可以直接配置了；

2）升级SQL  
Developer到最新版本：把原来“app\用户名\product\11.2.0\dbhome\_1”下的的删除，从官网下载最新版本，直接解压得到一个sqldeveloper文件夹放到同一位置即可。

**十二、Oracle运维**

概述

该文档主要目的是降低现场实施人员及用户Oracle数据库的管理难度，提高Oracle数据库技术能力，文档针对Oracle9i、10g两个版本提供了一套完整的Oracle数据库监控、管理的思路、方法步骤，依照该手册进行Oracle数据库的日常工作，能有效的把握Oracle后台数据库的整体运行健康状况，通过收集相关重要信息分析，能很好的防范即将出现的系统风险，系统出现问题后尽快的定位问题，现场解决一部分常规数据库问题。对其它专业要求比较强的数据库问题，也能为后续Oracle专家深入分析、诊断问题提供规范、完整的信息。

文档按问题处于的阶段分两部分－事前阶段、事中阶段，事前阶段描述了每天、每周末、每月末针对数据库所需进行的管理工作，如：日常监控，包括有**环境监控、数据库运行状况监控、性能监控**；日常数据库管理，包括：**系统运行快照采集、表空间管理、数据库备份恢复、表、索引统计分析、TOP会话、SQL执行计划**信息查看等。涉及有相关图形化管理工具使用方法、数据库自动脚本、命令使用方法。事中阶段描述了数据库发生问题时处理思路，需要收集哪些相关信息。

事前阶段

日常工作－每天应做工作内容

1、工作内容－日常环境监控

1.1系统运行环境监控

查看Oracle 数据文件、控制文件、联机日志及归档日志存放的文件系统或裸设备空间使用情况。

|  |
| --- |
| 重点关注Oracle软件及数据文件所在卷空间使用率：  su - oracle  AIX、linux查看磁盘空间：df –kv  HP-UX查看磁盘空间：bdf |

检测操作系统CPU、内存、交换区、I/O配置状况

|  |
| --- |
| AIX ：  CPU、内存、网络、IO、进程、页面交换：topas  Linux、HP-UX ：  CPU、内存、网络、IO、进程、页面交换： top |

1.2数据库运行状况监控

1.2.1 外部

检查Oracle实例核心后台进程是否都存在、状态是否正常

|  |
| --- |
| $ ps -ef|grep ora\_ |

查看数据库实例是否能正常连接、访问

|  |
| --- |
| SQL> select status from v$instance; |

监听是否正常

|  |
| --- |
| $ lsnrctl status |

1.2.2 内部

是否有表空间出现故障

|  |
| --- |
| SQL> select tablespace\_name,status from dba\_tablespaces; |

日志文件是否正常

|  |
| --- |
| SQL> Select \* from v$log;  SQL> Select \* from v$logfile; |

2、工作内容－日常性能监控

2.1 间隔一段时间使用操作系统top等工具监控系统资源动态运行状况

|  |
| --- |
| CPU、内存、网络、IO、进程、页面交换等主要活动监控：： top、topas、vmstat、iostat等 |

2.2间隔一段时间对数据库性能进行监控

2.2.1 Oracle 9i 图形工具－PerformanceManager监控顶层会话及顶层SQL

1． 打开OEM控制台，选中要监控的数据库。

2． 工具中选择 Diagnostic Pack－Performance Manager，也可直接选中Top Session或Top SQL。

Oracle9i 的Performance Manager工具监控内容主要有：内存的使用情况，IO情况，Oracle数据库进程情况，sql语句运行情况等，主界面如下：

可以通过顶层会话下钻获取到相关SQL执行计划等信息，也可以直接查看TopSql选项获取当前执行最频繁、消耗资源最多的SQL语句

 在数据页签下面列出了监控的选项列表，可以根据各类选项对SQL语句进行排序。

选中相关SQL语句，单击右键选择“下钻”到“解释计划”查看执行计划：

执行计划显示如下：

2.2.2 Oracle10g OEM工具监控顶层会话及获取SQL详细信息

登录Oracle10g OEM，选择性能－ 其它监视链接：顶级活动

点击顶级会话中的会话ID

点击SQL ID，查看该顶级会话中SQL的详细信息

点击**计划**标签，查看该SQL语句的详细执行计划

浏览该顶层会话对应SQL语句的详细信息

2.2.3 字符界面下Sql语句及用户进程信息采集

n  通过视图查看当前主要影响性能SQL语句

|  |
| --- |
| 语法模版  SELECT \* FROM   (SELECT hash\_value,address,substr(sql\_text,1,40) sql,       [list of columns], [list of derived values]     FROM [V$SQL or V$SQLXS or V$SQLAREA]    WHERE [list of threshold conditions for columns]    ORDER BY [list of ordering columns] DESC)  WHERE rownum <= [number of top SQL statements];  实际举例  SELECT \* FROM    (SELECT hash\_value,address,substr(sql\_text,1,40) sql,        buffer\_gets, executions, buffer\_gets/executions "Gets/Exec"     FROM V$SQLAREA     WHERE buffer\_gets > 100000 AND executions > 10    ORDER BY buffer\_gets DESC)  WHERE rownum <= 10; |

n  跟踪用户进程获取统计信息：

|  |
| --- |
| 获取要跟踪的用户进程  SQL> select sid,serial#,username from v$session;  开始跟踪－结束跟踪  Exec dbms\_system.set\_ev(9,437,10046,8,‘用户名');  Exec dbms\_system.set\_ev(9,437,10046,0,‘用户名');  生成的跟踪文件在user\_dump\_dest目录下  tkprof工具输出跟踪报表信息  tkprof /opt/oracle/admin/ytcw/udump/ytcw\_ora\_1026.trc  /opt/oracle/admin/ytcw/udump/ytcw\_ora\_1026.prf  aggregate=yes sys=no sort=fchela |

3、工作内容－日常数据库管理

 3.1一天内间隔一定时间运行

3.1.1检查警告日志文件中最新错误信息

|  |
| --- |
| Linux、Unix系列平台：  vi alertsid.log  输入：“/ORA－” 回车进行查找  Windows 平台下使用常用的文本编辑工具即可查看搜索警告日志文件中Oracle错误信息 |

3.1.2系统运行状况快照采集

每天根据实际情况，在以下三个阶段手工运行Statspack快照采集，输出快照报表：

l  正常工作压力下

l  每天业务最高峰期

l  特殊业务运行阶段

3.1.2.1 Oracle 自动化脚本方式快照采集

创建当前时间点快照

如需采集当前数据库运行状况快照，取20分钟间隔两次运行该脚本。

|  |
| --- |
| 自动执行statspack快照脚本：statspack\_auto\_exec.sh  #!/bin/sh  # creator: james\_jiang  # function:   produce statpack snapshot  echo "Auto Execute Statspack"  $ORACLE\_HOME/bin/sqlplus /nolog <<EOF  connect perfstat/perfstat  exec statspack.snap  echo "Auto execute statspack successfully!"  exit  EOF |

输出最近两个快照时间点之间的快照信息报表

|  |
| --- |
| **自动产生最近两个快照时间点统计信息快照脚本：statspack\_auto\_report.sh**  #!/bin/sh  # creator: james\_jiang  # function:get statpack report  echo " Auto create statspack snapshot!"  SQLPLUS=$ORACLE\_HOME/bin/sqlplus  LOGFILE=$ORACLE\_HOME/spreport.log  REPFILE=$ORACLE\_HOME/spreport.lst  $ORACLE\_HOME/bin/sqlplus -S perfstat/perfstat <<EOF  SET ECHO OFF  SET HEADING OFF  SET FEEDBACK OFF  SET PAGESIZE 0  SET LINESIZE 1000  SET TRIMSPOOL ON  SPOOL $LOGFILE  select SNAP\_ID from (select SNAP\_ID from stats\$snapshot where INSTANCE\_NUMBER=1 order by SNAP\_TIME desc) where rownum<3;  SPOOL OFF;  set echo on  set feedback on  set heading on  exit  EOF    line1=`tail -1 $LOGFILE`  line2=`head -1 $LOGFILE`  echo "line1 is"$line1  echo "line2 is"$line2  $ORACLE\_HOME/bin/sqlplus -S perfstat/perfstat <<EOF  define begin\_snap=$line1  define end\_snap=$line2  define report\_name=$REPFILE  @?/rdbms/admin/spreport.sql  echo "Auto create statspack snapshot successfully!"  exit  EOF |

3.1.2.2 Oracle OEM图形管理工具实现系统快照采集

自动化脚本执行快照收集主要是Oracle版本的使用方式，OracleOEM图形工具自动执行快照采集，缺省1小时收集一次，可以根据实际情况修改收集间隔时间、降低对系统性能影响。

系统快照自动收集时间、间隔、保留期限设置

登录Oracle OEM，选择管理－ 自动工作量档案库

点击编辑，查看或修改快照收集时间及间隔

Oracle缺省系统快照每隔一小时执行一次，保留最近15天的所有快照，可根据实际情况修改调整，点确定后保存所做修改。

创建当前时间点系统快照

点击管理快照和保留的快照集下面的当前**快照ID**

选择创建保留快照集，点击创建

选择“是”开始执行快照创建

快照在当前时间点成功创建。

输出两个快照时间点之间的快照信息报表

修改原来“创建保留的快照集”为“查看报告”，选择起始快照号，点击创建

选择结束快照号，点击“确定”

开始创建两个快照时间点之间的所有统计信息报告

另存该输出快照报告为HTML文件

注：生成的统计信息快照报告放在专门目录下，定期对其整理、分析，作为EAS数据库运行整体状况及问题诊断的依据。

3.2 每天工作结束后、系统空闲时运行

3.2.1表空间使用率

3.2.1.1 SQL脚本方式查看

|  |
| --- |
| 脚本：FREESPACE.SQL  SELECT a.tablespace\_name, ROUND (100 - b.free / a.total \* 100) used\_pct,         ROUND (a.total / 1024 / 1024) "total(MB)",         ROUND (b.free / 1024 / 1024) "free\_total(MB)",         ROUND (b.max\_free / 1024 / 1024) "free\_max(MB)", b.free\_cnt fragment    FROM (SELECT   tablespace\_name, SUM (BYTES) total              FROM dba\_data\_files          GROUP BY tablespace\_name) a,         (SELECT   tablespace\_name, SUM (BYTES) free, MAX (BYTES) max\_free,                   COUNT (BYTES) free\_cnt              FROM dba\_free\_space          GROUP BY tablespace\_name) b  WHERE a.tablespace\_name=b.tablespace\_name |

3.2.1.2 图形界面查看表空间使用率

Oracle9i OEM 表空间管理

Oracle10g OEM表空间管理

登录OEM后选择管理－ 表空间

n

3.2.2数据库备份及日志清理

数据库备份主要提供两种方式，物理备份及逻辑备份，物理备份主要使用OracleRMAN工具，逻辑备份主要使用Oracle导出工具Exp及Expdp。

Oracle 物理备份(RMAN)

该方式下周一到周五每天做一次增量备份，并检查备份是否正确，同时清理归档日志。

3.2.2.1 Oracle RMAN自动化脚本增量备份

|  |
| --- |
| **RMAN备份环境初始化设置：**  rman nocatalog  rman>connect target sys/oracle  rman>configure controlfile autobackup on;  rman>configure controlfile autobackup format for device type disk to **'f:\rman\_bak\%F.ctl'**;  rman>configure snapshot controlfile name to **'f:\rman\_bak\snap\_%F.ctl'**; |

|  |
| --- |
| Rman备份命令写到一个脚本中，在命令行中执行这个脚本  **RMAN TARGET / NOCATALOG sys/oracle CMDFILE f:\ backup\_incre\_1.rcv LOG f:\ backup\_incre\_1.log** |

该命令可设置为crontab（unix/linux），bat批处理任务（Windows），在每天特定的时间点自动运行。

|  |
| --- |
| **增量备份脚本： backup\_incre\_1.rcv**  run  {allocate channel c1 type DISK ;  backup incremental level = 1 format **'f:\rman\_bak\incre\_1\_%d\_%s\_%p.bak'**  (database include current controlfile);  backup format **'f:\rman\_bak\arch%u\_%s\_%p.bak'**  (archivelog from time 'sysdate-1' all delete input);  } |

|  |
| --- |
| **显示RMAN备份集信息：**  RMAN> list backupset; |

**注：上述脚本中涉及的文件路径需根据现场环境具体情况进行相应修改。**

3.2.2.2 Oracle10gOEM图形化方式设置RMAN备份自动执行任务

RMAN备份主要参数设置

登录10g OEM，选择维护 － 备份设置

**磁盘设备备份路径、并行度指定（根据服务器cpu个数匹配）**

**备份策略设置**

主要是指定备份磁盘位置及备份保留时间，下图设置中为周备份策略考虑，保留最近7天备份。

两种RMAN自动调度备份策略

1)    使用Oracle建议的自动调度备份策略

选择维护－调度备份

选择“调度Oracle建议的备份”

选择备份目标介质，缺省备份到磁盘介质

该备份策略内容描述

设置该策略执行数据库全量或增量备份的调度时间

2)    自定义RMAN自动备份策略

选择“调度定制备份”

选择备份类型、模式、归档日志、过时备份清理策略

备份目标介质设置（缺省使用前面“备份设置”中设置的参数）

备份自动执行的调度时间设置（通常设置在系统空闲时进行，如晚上12点过后）

备份设置信息复查，确定后提交作业

通过查看作业可以了解备份任务进展情况。

3)    查看当前数据库RMAN备份信息

选择维护－备份/恢复中“备份报告”可以查看所有备份执行情况

注：前面描述的是数据库全量备份的调度策略设置，增量备份的调度策略设置方法类似，只需要在备份类型中选择增量备份即可。设置完成后EAS数据库自动备份策略为一周一次全量备份，在星期六晚上12点进行。周一到周五每天晚上12点执行一次增量备份，保留最近7天备份，过期备份、归档日志自动删除。

Oracle逻辑备份（EXP/EXPDP）

1) 操作系统级设置自动备份任务

|  |
| --- |
| root用户登录操作系统，运行crontab –e，添加以下内容：  50 23 \* \* 1-6  su – oracle -c /usr/app/oracle/expdp/eas\_expdp\_MontoSat.sh > /dev/null 2>&1  **#星期一到星期六晚上11点50导出数据**  50 23 \* \* 0  su – oracle -c /usr/app/oracle/expdp/eas\_expdp\_Sunday.sh > /dev/null 2>&1  **#星期日11点50开始导出数据，并删除上周一到周六导出的备份数据。** |

2) 逻辑备份脚本

逻辑备份脚本 eas\_expdp\_MontoSat.sh（星期一到星期六）

|  |
| --- |
| #!/bin/sh  #====================================================  # SCRIPT    : eas\_expdp\_MontoSat.sh  # AUTHOR    : James\_jiang  # Date      : 2007-10-10  # REV       : 1.0  # PLATFORM  : AIX Linux Solaris HpUnix  # PURPOSE   : This script is used to run logic backup.  # Copyright(c) 2007 Kingdee Co.,Ltd.  # All Rights Reserved  #=====================================================  DAY=`date +%a`  FILE\_TARGET=eas\_expdp\_`expr $DAY'.dmp  FILE\_LOG=eas\_expdp\_`expr $DAY`.log  export FILE\_TARGET FILE\_LOG  expdp salhr/salhr schemas=salhr directory=eas\_expdp\_dir dumpfile=$FILE\_TARGET logfile=$FILE\_LOG job\_name=cases\_export parallel=4 |

逻辑备份脚本 eas\_expdp\_Sunday.sh（星期日）

|  |
| --- |
| sqlplus "/as sysdba"  SQL>create directory eas\_expdp\_dir as '/usr/app/oracle/expdp';  SQL>grant read,write on directory eas\_expdp\_dir to salhr; |

Oracle10g 版本逻辑备份使用expdp工具，在部署导出自动任务前需创建导出dmp文件存放目录并授权给EAS数据库用户：

|  |
| --- |
| #!/bin/sh  #=====================================================  # SCRIPT    : eas\_expdp\_Sunday.sh  # AUTHOR    : James\_jiang  # Date      : 2007-10-10  # REV       : 1.0  # PLATFORM  : AIX Linux Solaris HpUnix  # PURPOSE   : This script is used to run logic backup.  #=====================================================  DAY=`date +%a`  FILE\_TARGET=eas\_expdp\_`expr $DAY`.dmp  FILE\_LOG=eas\_expdp\_`expr $DAY`.log  export FILE\_TARGET FILE\_LOG  cd /usr/app/oracle/expdp  rm -f eas\_expdp\*.dmp eas\_expdp\*.log  expdp salhr/salhr schemas=salhr directory=eas\_expdp\_dir dumpfile=$FILE\_TARGET logfile=$FILE\_LOG job\_name=cases\_export parallel=4 |

附： 上述脚本中导出路径 /usr/app/oracle/expdp需根据现场实际情况修改，备份脚本eas\_expdp\_MontoSat.sh、eas\_expdp\_Sunday.sh需赋予可执行权限。Expdp导出目录需要创建，具体方法请参看本文档 四 日常工作－数据库第一次安装部署后需做的工作

3.2.3根据监控信息，对需要的表、索引统计分析

3.2.3.1 Oracle9i 自动化脚本方式对表、索引进行统计分析

Oracle9i缺省不对表进行改动监控，如果需要根据监控信息来判断是否需对表进行重新统计分析，则需要手工打开表监控开关，如下：

|  |
| --- |
| **打开表监控开关：**  alter table 表名 monitoring；//监控表信息记录在sys.dba\_tab\_modifications视图中 |

|  |
| --- |
| **根据监控信息对用户统计信息分析收集脚本：**  dbms\_stats.gather\_schema\_stats(ownname=>'nmeas',options=>'GATHER AUTO') |

3.2.3.2 Oracle10g 自动化任务表、索引统计分析方式

Oracle10g 缺省自动对所有表变动进行监控，并自动执行所有用户统计分析，可以禁止该缺省方式，在dbconsole中手工配置对特定用户统计分析任务。

|  |
| --- |
| Oracle10g禁用、启用缺省数据库自动统计信息分析收集任务：  exec dbms\_scheduler.disable('SYS.GATHER\_STATS\_JOB');  exec dbms\_scheduler.enable('SYS.GATHER\_STATS\_JOB'); |

Oracle10g OEM图形工具自定义配置统计分析任务

登录OEM，选择管理－统计信息管理－管理优化程序统计信息

选择“操作”－搜集优化程序统计信息

选择“方案”，点击下一步

设置用户统计信息分析自动执行的时间调度

示例中设置的是从2007-03-29开始，每天晚上12点自动执行用户NMEAS统计信息分析，

用户统计信息分析任务设置完整栏目显示

提交完成任务设置

查看数据库中所有自动调度任务执行情况

OEM中选择栏目：管理－统计信息管理（管理优化程序统计信息）

选择相关链接－作业调度程序

注：上述配置的自动统计分析任务利用Oracle自动监控来判断哪些对象改动较大，需要重新进行统计分析。然后在调度的时间对其重新进行统计分析。

日常工作－每隔一周工作内容

1.文件整理工作

n  警告日志、跟踪文件、dump文件清理

n  备份文件整理

n  Statspack统计分析报告整理

2.数据库全量备份

2.1 Oracle 9i RMAN自动化脚本方式全量备份

|  |
| --- |
| Rman备份命令写到一个脚本中，在命令行中执行这个脚本  **RMAN TARGET / NOCATALOG sys/oracle CMDFILE e:\** backup\_incre\_0.rcv **LOG e:\** backup\_incre\_0**.log** |

|  |
| --- |
| 数据库完全备份脚本： backup\_full\_0.rcv  run  {allocate channel c1 type DISK ;  backup incremental level = 0 format 'f:\rman\_bak\incre\_0\_%d\_%s\_%p.bak'  (database include current controlfile);  backup format 'f:\rman\_bak\arch%u\_%s\_%p.bak'  (archivelog from time 'sysdate-1' all delete input);  } |

|  |
| --- |
| 删除过期备份：  RMAN>delete expired backup; |

2.2 Oracle 10g OEM 图形方式创建RMAN全量备份数据库任务

具体方法、步骤请参看数据库日常工作－每天应做工作内容中RMAN备份部分

3.根据一周数据增长率分析预留数据文件下一周所需增长空间

3.1 SQL脚本方式查看

|  |
| --- |
| 数据文件空间使用率查看脚本：  SELECT   df.tablespace\_name, ROUND (df.BYTES / 1024 / 1024) "total(MB)",           ROUND ((df.BYTES - x.free) / 1024 / 1024) "used(MB)",           ROUND (x.hw / 1024 / 1024) hwatermarker,           ROUND (x.free / 1024 / 1024) "free(MB)", df.file\_name      FROM dba\_data\_files df,           (SELECT   file\_id, MAX (block\_id \* p.VALUE) hw, SUM (BYTES) free                FROM dba\_free\_space, v$parameter p               WHERE p.NAME = 'db\_block\_size'            GROUP BY file\_id) x     WHERE x.file\_id = df.file\_id  ORDER BY 1, 2 |

3.2 Oracle 9i OEM 数据文件管理

3.3 Oracle 10g OEM 数据文件管理

4.索引使用情况及碎片分析

4.1表包含的索引及相关列检查

|  |
| --- |
| 1、  表及索引创建、修改日期检查  EAS用户登录到数据库，执行语句：  SQL>SELECT OBJECT\_NAME,OBJECT\_TYPE,CREATED,LAST\_DDL\_TIME,STATUS FROM USER\_OBJECTS；  2、  表包含的索引及索引相关列检查  SQL>  SELECT INDEX\_NAME,TABLE\_NAME,COLUMN\_NAME  FROM USER\_IND\_COLUMNS WHERE TABLE\_NAME LIKE 'T\_%' ORDER BY TABLE\_NAME；  3、  特定表及其关联索引、列详细信息检查  SELECT A.TABLE\_NAME,A.INDEX\_NAME,COLUMN\_NAME,CREATED ,LAST\_DDL\_TIME,LAST\_ANALYZED FROM USER\_IND\_COLUMNS A,USER\_OBJECTS  B,USER\_INDEXES C WHERE A.INDEX\_NAME=B.OBJECT\_NAME AND B.OBJECT\_NAME=C.INDEX\_NAME AND C.TABLE\_NAME IN（'',''）  注：如需输出语句执行结果信息，可在语句执行前spool tableindex.list,执行完成后再spool off； |

4.2自动化脚本方式对索引进行碎片分析

|  |
| --- |
| 每周监测一次索引的碎片情况，根据情况制定索引的重建频率以提高索引使用效率。  1、产生EAS用户分析索引的脚本：  SQL>select 'analyze index ' || index\_name ||' VALIDATE STRUCTURE' from user\_indexes;  2、执行EAS用户下所有索引分析：  SQL> analyze index <index\_name> VALIDATE STRUCTURE;  ……..  3、基于分析结果，查看索引碎片率：  SQL> select name,del\_lf\_rows\_len,lf\_rows\_len,(del\_lf\_rows\_len/lf\_rows\_len)\*100 from index\_stats;  索引碎片率(%) = (被删除的索引长度/索引总长)\*100  4、对碎片率高的索引执行重建整理  SQL> alter index <索引名> rebuild; |

4.3打开索引自动监控开关

如数据库中新增加、修改了索引，则可以打开这些索引的自动监控。

|  |
| --- |
| 监测索引的使用情况，根据使用情况，删除未使用的索引，并添加能提高查询和处理性能的索引。  SQL> alter index <索引名> monitoring usage;  SQL> alter index <索引名> nomonitoring usage;  SQL> select index\_name,used from v$object\_usage; |

5.对用户所有表、索引进行统计分析

5.1 查询EAS用户所有表、索引的最新统计分析时间

|  |
| --- |
| 表：  SQL>select table\_name,last\_analyzed from user\_tables order by 2;  索引：  SQL>select table\_name,index\_name,last\_analyzed from user\_indexes order by 1,3  注：last\_analyzed字段显示的日期太老，则表明该表或索引最近未做统计分析，如果恰好最近大量更新、导入或删除了记录，需要重新对其执行统计分析。 |

5.2 自动化脚本方式对所有表、索引统计分析

|  |
| --- |
| 对用户所有对象进行完整统计信息分析收集脚本：  dbms\_stats.gather\_schema\_stats(ownname=>'nmeas',method\_opt=>'FOR ALL INDEXED COLUMNS SIZE auto',estimate\_percent =>100,degree=>DBMS\_STATS.DEFAULT\_DEGREE,cascade=>true) |

 9i需（加入crontab或bat任务）

5.3 Oracle 10g OEM图形化自定义对所有表、索引统计分析的自动化调度任务

登录OEM，选择管理－统计信息管理－管理优化程序统计信息－“操作”－搜集优化程序统计信息

范围选项方案中选择“定制选项”

添加用来做统计信息收集的EAS用户

配置周统计信息收集Oracle相关参数

配置周统计信息收集任务执行的调度时间（下图中配置的为每周星期六晚上12点自动执行统计信息搜集）

浏览周统计信息收集任务配置的完整信息

提交完成周统计信息收集任务配置

注： 每周末EAS用户下所有表、索引通过上述任务执行完整的统计分析。

6.导出表、索引最新统计分析数据

将导出的统计分析数据导入测试库，可以在测试库重新构建性能关键或所有sql语句执行计划、与基准执行计划进行比较。

|  |
| --- |
| 导出用户当前统计信息：  exec dbms\_stats.export\_schema\_stats('nmeas','stats\_export')  注：存储导出信息的表stats\_export需在安装部署EAS后台数据库时创建。 |

可以考虑与周统计信息收集放在同一job中

7.性能报告分析

就一周来的statspack报告进行整理、分析，主要关注：

n  数据库整体性能状况指标

n  数据库主要等待事件

n  最消耗cpu、内存资源、I/O的SQL语句

日常工作－每月应做工作内容

1. 性能全面分析

全面分析一次STATSPACK报告

n  数据库主要性能指标

n  数据库主要等待事件

n  最消耗cpu、内存资源、I/O的SQL语句

 空间使用增长的全面分析

n  确定是否需要扩充存储空间

n  考虑预留足够下个月使用的空间

2. 备份数据转备

将一个月以来的Rman备份文件打包，转存到外部存储介质，有条件最好存放异地。

 日常工作－数据库第一次安装部署后需做的工作

1.Statspack－系统快照采集工具初始化

Oracle  自动化脚本方式初始化Statspack工具

|  |
| --- |
| **安装statspack工具**  SQL> @?/rdbms/admin/spcreate  创建用户perfstat  要求输入口令：perfstat  要求输入表空间：users  及临时表空间：temp  ……创建完成。 |

**注：Oracle 10g OEM图形化系统快照采集不用进行Statspack初始化**

2.创建统计信息导出表

|  |
| --- |
| 创建用户统计信息导出表：  exec dbms\_stats.create\_stat\_table('nmeas','stats\_export') |

3.运行EAS用户下所有表、索引统计分析，导出基准统计信息

|  |
| --- |
| 对用户所有对象进行完整统计信息分析收集脚本：  dbms\_stats.gather\_schema\_stats(ownname=>'nmeas',method\_opt=>'FOR ALL INDEXED SIZE auto'，estimate\_percent=>dbms\_stats.auto\_sample\_size,degree=>DBMS\_STATS.DEFAULT\_DEGREE,cascade=>true) |

|  |
| --- |
| 导出用户当前统计信息：  exec dbms\_stats.export\_schema\_stats('nmeas','stats\_export') |

4.创建Oracle逻辑备份dump文件存放目录

|  |
| --- |
| **Oracle用户登录操作系统，执行：**  sqlplus "/as sysdba"  SQL>create directory eas\_expdp\_dir as '/usr/app/oracle/expdp';  SQL>grant read,write on directory eas\_expdp\_dir to eas用户名;    注：目录/usr/app/oracle/expdp根据实际环境修改。 |

事中阶段

既使按规范操作手册对数据库进行日常监控、管理，也不能避免数据库运行过程中产生各式各样的问题，这些问题涉及的原因方方面面，因此，在出现问题的事中阶段关键是收集、掌握问题发生时所有相关信息，并对其进行分析，准确的定位问题，找出最好的解决办法。

Oracle数据库出现问题时需掌握的相关信息

1.问题症状描述

返回的错误代码及描述信息：

|  |
| --- |
| l  EAS应用返回“ORA-”错误信息  l  警告日志文件－Alertsid.log出现“ORA-”错误信息 |

2.问题在什么地方出现

|  |
| --- |
| l  安装db的哪一步  l  备份、恢复到哪一步报错  l  应用程序运行时报错  l  应用程序连接报错  l  数据库正常启动、关闭报错  l  数据库正常使用报错（日志文件） |

3.问题在什么时间出现

|  |
| --- |
| l  一段时间内持续出现  l  某个特定时间点出现 |

4.问题在什么条件下出现

|  |
| --- |
| l  硬、软件升级，更新补丁后  l  批处理作业在运行导致  l  操作系统存储进行改动  l  反病毒软件运行  l  业务高峰期(并发用户数多少，此时是否有大量用户在做报表、计算等复杂业务) |

5.问题涉及的范围

|  |
| --- |
| l  个别系统或区域  l  相关的所有系统或区域 |

6.问题是否能重现

7.数据库运行环境软、硬件基本信息

|  |
| --- |
| l  操作系统平台版本、补丁号  l  数据库版本、补丁号  l  第三方软件版本、补丁号  l  Cpu、内存、交换区配置  l  存储配置及空间使用率 |

8.Oracle性能相关

如出现Oracle数据库整体性能下降、某功能响应时间过长甚至没反应等性能问题，除了需了解上述的信息外，还需要进行额外信息收集：

|  |
| --- |
| l  Oracle数据库这段时间系统运行快照报告  l  定位该功能主要sql后，导出及相关表、索引结构及统计数据 |