Oracle

## 一 基础

数据操作语言 select insert update delete merge

数据定义语言 create alter drop truncate

数据控制语言 grant revoke

事务定义语言 commit rollback savepoint

Oracle 自动建表，并将表中的数据自动拷贝到另一张表中

Create table A as select \* from table B where B.no=’’;

表之间数据拷贝

Insert into table A (f1,f2,f3…) as select v1,v2,v3 … from table B;

Oracle表合并

Merge into produce a

Using (select ‘1717’ as num ,’01’ as reqno ) b

On (a.num=b.num and a.reqno=b.reqno)

When matched then

Update set product\_name=’’,category=’’

When not matched then

Insert into (num,reqno) values(‘1717’,’01’);

Oracle递归分层查询查询

Start with (从某个节点id开始)

Connect by prior (子节点id和父节点pid)

Select \* from

Emp

Start with empno=7368

Connect by prior mgr=empno;

可以递归的进行树的查询，从上往下查询，或者从下往上查询

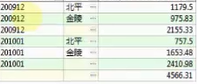
Connect by prior pid=id(往上)或者id=pid(往下)就可以实现oracle的递归查询。

分析函数rollup分组与小记函数

Select earnmoonth,area,sum(personincome)

From earnings

Group by rollup(earnmonth,area);



Select earnmonth,

(case when (grouping(area)=1) and (grouping(earnmonth)=0) then ‘月份小计’

When (grouping(area)=1) and (grouping(earnmonth)=1) then ‘总计’

Else area end) as area,

Sum(personincome)

From earnings

Group by rollup(earnmonth,area)



Select earnmonth 月份,area 地区,sname 姓名,personincome 收入,

rank() over (partition by earnmonth,area order by personinome desc) 排名

From earnings;



Select earnmonth 月份,area 地区,sname 姓名,personincome 收入,

Dense\_rank() over (partition by earnmonth,area order by personinome desc) 排名

From earnings;



Select earnmonth 月份,area 地区,sname 姓名,personincome 收入,

Row\_number() over (partition by earnmonth,area order by personinome desc) 排名

From earnings;

连续求和

Select

earnmonth 月份,

area 地区,

sname 姓名,

personincome 收入,

sum(personincome) over (partition by earnmonth,area order by personinome asc) 总收入

From earnings;

Select distinct earnmonth 月份,area 地区,

Max(personincome) over (partition by earnmonth,area) 最高值,

min(personincome) over (partition by earnmonth,area) 最低值,

avg(personincome) over (partition by earnmonth,area) 平均值,

sum(personincome) over (partition by earnmonth,area) 总额

from earnings;

//产看事务

select \* from v$transavtion;

//查看锁

Select \* from v$lock;

试探锁的方式:

Select \* from emp1 where empno=7782 for update nowait;

Select \* from emp1 where empno=7782 for update wait 5;

Select \* from emp1 where job =’clerk’ for update skip locked;

把数据库session连接kill掉

首先先查看v$lock表，看sid哪个占用了太长的时间，在通过sid去v$session

里面找到它，进行kill操作。

Select sid,serial# from v$session where sid=170;

Alter system kill session ‘sid,serial’;

创建索引的原则：

该表主要用来根据指定条件查询，update操作相对较少，比如报表，图形报表，这些表是通过聚合其他普通表的数据进行拼装的，主要提供根据条件进行查询，数据量大的情况下我们可以对查询条件进行创建索引。

//查询索引

Select \* from user\_ind\_columns;

//索引碎片分析整理

Analyze index ind\_1 validate structure。

//查询碎片率，看是否需要对碎片进行整理

Select name,height.pct\_used,del\_lf\_rows/lf\_rows from index\_stats;

//delete t where rownum<700000 此时会产生索引碎片

我们需要对索引碎片进行整理

Alter index ind\_1 rebuild online;

Select \* from user\_tables;

创建物化视图进行全表刷新

Create materialized view v\_as

Refresh force on commit

As

Select

a.id,

a.name,

b.clsid,

b.name as clsname

from a,b where a.clsid=b.clsid;

生成本地快照，全量刷新。

Create materialized viewmv\_ab

Refresh force on demand

Start with sysdate

Next sysdate+1

As

Select A.ID,A.NAME,B.CLSID,B.NAME as clsname

From A,B

Where A.clsid=B.clsid;

定时间隔手工刷新。

快速刷新

Create materialized viewmv\_ab

Refresh fast on demand

Start with sysdate

Next sysdate+1/1440

As

Select A.ID, A.NAME, B.CLSID, B.NAME as clsname

From A,B

Where A.clsid=B.clsid;

同义词:同一张表的第二种描述，也就是一张表的虚表。

然后授权另一个用户可以查询另一张虚表。这样就可以跨

用户取数了

Grant create synonym to scott;

Create synonym abc for emp;

Create public synonym to scott;

Create public synonym xyz for emp;

Drop public synonym xyz;

OLTP和OLAP

在互联网时代，海量数据存储与访问成为系统设计与使用的瓶颈对海量数据处理，按照使用场景，主要分为两类：联机事务处理(OLTP)和联机分析处理(OLAP).

联机事务处理：面向交易的处理系统，其基本特征是原始数据可以立即传送到计算中心进行处理，并在很短的时间内给出结果

联机分析处理：通过多维的方式对数据进行分析，查询和报表，可以同数据挖掘工具，统计分析工具配合使用，增强决策分析能力。

分区

Create table sale(produce\_id varchar2(5),sale\_count number(10,2)){

Partition p1 range(sale\_count){

Partition p1 values less than(1000),

Partition p2 values less than(2000),

Partition p3 values less than(3000)

}

}

查看分区：

Select \* from user\_tab\_partitions;

查询分区中的数据

Select \* from sale partition(p1);

Select \* from sale partition(p2);

Select \* from sale partition(p3);

添加分区

Alter table sale ass partition p4 values less than(maxvalue);

如果要更新分区的话

Update sale set sale\_count=1500 where product\_id =1;

创建分区索引

Create index idx\_count on sale(sale\_count) local;

间隔分区

Interval Partitioning 实际是range分区的引申