【济南中心】JavaEE就业班同步笔记第二阶段：Oracle-part03

小鲁哥哥 • 2017-5-20 20:23:58

【济南中心】JavaEE就业班同步笔记第二阶段：Oracle-part03

第一节

1.1 视图概述

1.2.1知识概述

1、简介

视图是基于一个表或多个表或视图的逻辑表，本身不包含数据，通过它可以对表里面的数据进行查询和修改。视图基于的表称为基表。视图是存储在数据字典里的一条select语句。 通过创建视图可以提取数据的逻辑上的集合或组合。

2、优点：

简化数据操作：视图可以简化用户处理的方式；

着重于特定数据：不必要的或敏感的数据可以不出现在视图中；

视图提供了一个简单而有效的安全机制（只读），可以定制不同用户对数据访问；

提供向后兼容性：视图使用户能够在表的结构构更改时为表创建向后兼容接口。

对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。

用户通过简单的查询可以从复杂查询中得到结果。

维护数据的独立性，试图可从多个表检索数据。

对于相同的数据可产生不同的视图。

1.3.1 知识概述

1、创建和修改视图

CREATE [OR REPLACE] [FORCE] VIEW view_name

AS subquery

[WITH CHECK OPTION]
[WITH READ ONLY]
OR REPLACE：如果视图已存在，则重新创建视图，相当于修改视图；
FORCE：不管基表是否存在，都会创建视图；
subquery ： select 查询语句（可以是单表，可以使多表，可以包含子查询）；
WITH CHECK OPTION：插入或修改的数据必须满足视图定义的约束；
WITH READ ONLY：该视图只能查询，不能修改或者插入数据。
2、删除视图
DROP VIEW view_name
1.4 视图-简单视图
1.4.1 知识概述
简单视图:subquery子句是一个单表查询语句。
创建：
CREATE OR REPLACE VIEW VIEW_OWNERS
AS SELECT * FROM T_OWNERS WHERE OWNERTYPEID=1;
查询视图数据：SELECT * FROM VIEW_OWNERS [WHERE OWNERTYPEID=1];
修改视图数据：UPDATE VIEW_OWNERS SET NAME='Oracle' WHERE ID=2;
删除视图数据: DELETE FROM VIEW_OWNERS WHERE ID=2;
注意：不建议在视图上插入、修改、删除数据
1.5 视图-带检查约束的视图
1.5.1 知识概述
带检查约束的视图：使用WITH CHECK OPTION关键字，
create or replace view view_address2 as
select * from T_ADDRESS where areaid=2
with check option;
使用[WITH CHECK OPTION]关键字后，在插入、修改数据时会检查是否满足subquery子句中的where条件，只有满足才能执行保存到数据库中。

第二节
2.1 视图-只读视图
2.1.1知识概述
使用关键字WITH READ ONLY,此时只能查询视图中的数据，不能插入、修改和删除操作。

2.2 视图-带错误的视图

2.2.1知识概述

使用关键字FORCE，意思是强制执行创建视图的语句，忽略表不存在的错误

2.3 视图-多表关联的视图

2.3.1知识概述

2.3.1.1 创建

其中的subquery子句是一个多表关联的select查询语句。

2.3.1.2 键保留表

官方介绍：

A table is key preserved if every key of the table can also be a key of the result of the join.
It is not necessary that the key or keys of a table be selected for it to be key preserved.
It is sufficient that if the key or keys were selected, then they would also be key(s) of the result of the join.

键保留表的理解是：

一个复杂视图，若需要出现键保留表的话则必须保证基表中至少有一张表是有主键的！(如果两个没有主键表进行关联时是不会出现键保留表的；另外如果视图中有一张基表具有主键，就一定会出现另一张基表成为键保留表的现象)

其次，这两张表在进行关联时(可以是表连接也可以是多表查询，但一定要有关联条件，其关联条件其实相当于两表的主外键关系)，如果关联条件是使用了主键的话，则外键表为键保留表。

2.4 视图案例-聚合统计的视图

2.4.1 知识概述

该类型的subquery子句是一个分组统计或者排序的select查询语句

2.5 物化视图-概述和语法

2.5.1 知识概述

2.5.1.1 概述

物化视图 （Materialized View）在9i以前的版本叫做快照（SNAPSHOT），从9i开始改名叫做物化视图。它是用于预先计算并保存表连接或聚集等耗时较多的操作的结果，这样，在执行查询时，就可以避免进行这些耗时的操作，从而快速的得到结果。物化视图有很多方面和索引很相似：使用物化视图的目的是为了提高查询性能；物化视图对应用透明，增加和删除物化视图不会影响应用程序中SQL 语句的正确性和有效性；物化视图需要占用存储空间；当基表发生变化时，物化视图也应当刷新。

与视图的区别是

物化视图和视图类似，反映的是某个查询的结果，但是和视图仅保存SQL定义不同，物化视图本身会存储数据，因此是物化了的视图。

物化视图的优缺点

优点：

1，物化视图的最大的优势是可以提高性能：Oracle的物化视图提供了强大的功能，可以用于预先计算并保存表连接或聚集等耗时较多的操作的结果，这样，在执行查询时，就可以避免进行这些耗时的操作，而从快速的得到结果。

2， 物化视图有很多方面和索引很相似

3，通过预先计算好答案存储起来，可以大大地减少机器的负载

A，更少的物理读--扫描更少的数据

B，更少的写--不用经常排序和聚集

C。减少CPU的消耗--不用对数据进行聚集计算和函数调用

D，显著地加快响应时间--在使用物化视图查询数据时(与主表相反)，将会很快的返回查询结果

缺点：

1，物化视图用于只读或者"精读"环境下工作最好，不用于联机事务处理系统(OLTP)环境，在事实表等更新时会导致物化视图行锁，从而影响系统并发性。

2，物化视图有出现无法快速刷新，导致查询数据不准确的现象

3，Rowid物化视图（创建的物化视图通常情况下有主键，rowid,和子查询视图）只有一个单一的主表，不能包括下面任何一项:

A，Distinct 或者聚合函数.

B，Group by，子查询，连接和SET操作

4，物化视图会增加对磁盘资源的需求，即需要永久分配的硬盘空间给物化视图来存储数据

5，物化视图的工作原理受一些可能的约束，比如主键，外键等。

2.5.1.2 语法

BUILD IMMEDIATE 是在创建物化视图的时候就生成数据

BUILD DEFERRED 则在创建时不生成数据，以后根据需要再生成数据。 默认为BUILD IMMEDIATE。

REFRESH方式：指当基表发生了DML操作后，物化视图何时采用哪种方式和基表进行同步。

FAST刷新采用增量刷新，只刷新自上次刷新以后进行的修改。

COMPLETE刷新对整个物化视图进行完全的刷新。

FORCE方式，则Oracle在刷新时会去判断是否可以进行快速刷新，如果可以则采用FAST方

式，否则采用COMPLETE的方式。FORCE是默认的方式。

刷新的模式有两种：ON DEMAND和ON COMMIT。ON DEMAND指需要手动刷新物化视图（默认）。ON COMMIT 指在基表发生COMMIT操作时自动刷新

第三节

3.1 物化视图-手动刷新的物化视图

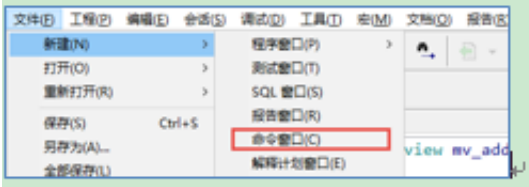
3.1.1知识概述

创建视图：

```
create materialized view mv_address
as
select ad.id,ad.name adname,ar.name ar_name
from t_address ad,t_area ar
where ad.areaid=ar.id
```

手动刷新物化视图：

```
begin
DBMS_MVIEW.refresh('MV_ADDRESS','C');
end;
```




```
SQL> EXEC DBMS_MVIEW.refresh('MV_ADDRESS','C');
```

3.2 视图案例-自动刷新的视图

Refresh on commit 在数据提交时刷新物化视图

3.3 物化视图-创建时不生成数据

3.3.1知识概述

创建时不生成数据，在数据提交时会刷新物化视图，也可以手动刷新

3.4 物化视图-增量刷新物化视图

3.4.1 知识概述

3.4.1.1 创建增量刷新的物化视图

1、创建物化视图日志

创建出来的日志的名称为【mlog$_表名称】

2、创建物化视图

注意：创建增量刷新的物化视图，必须：
1. 创建物化视图中涉及表的物化视图日志。
2. 在查询语句中，必须包含所有表的rowid（以rowid方式建立物化视图日志）
3.4.1.2 物化视图日志

```
create materialized view mv_address3
build deferred
refresh
on commit
as
select ad.id,ad.name adname,ar.name ar_name
from t_address ad,t_area ar
where ad.areaid=ar.id;
```

SNAPTIME$$：用于表示刷新时间。
DMLTYPE$$：用于表示DML操作类型，I表示INSERT，D表示DELETE，U表示UPDATE。
OLD_NEW$$：用于表示这个值是新值还是旧值。N（EW）表示新值，O（LD）表示旧值，U表示UPDATE操作。
CHANGE_VECTOR$$：表示修改矢量，用来表示被修改的是哪个或哪几个字段。
此列是RAW类型，其实Oracle采用的方式就是用每个BIT位去映射一个列。
插入操作显示为：FE, 删除显示为：OO更新操作则根据更新字段的位置而显示不同的值。

第四节
4.1 序列概述
4.1.1知识概述
序列是oacle提供的用于产生一系列唯一数字的数据库对象。序列的用途一般用来填充主键和计数。
简单的序列
4.2 序列-语法
4.2.1知识概述

```
CREATE SEQUENCE sequence  --创建序列名称
       [INCREMENT BY n]  --递增的序列值是n 如果n是正数就递增,如果是负数就递减 默认是1
       [START WITH n]    --开始的值,递增默认是minvalue 递减是maxvalue
       [{MAXVALUE n | NOMAXVALUE}] --最大值
       [{MINVALUE n | NOMINVALUE}] --最小值
       [{CYCLE | NOCYCLE}] --循环/不循环
       [{CACHE n | NOCACHE}];--分配并存入到内存中
```

4.3 创建、修改、删除序列

4.3.1知识概述

1、有最大值的非循环序列

注意：起始值不能比最小值小

2、有最大值的循环序列

注意：循环序列必须指定最大值

3、带缓存的序列

注意：设置缓存的值必须小于循环的值的个数

修改和删除序列

第五节

5.1 同义词概述和语法

5.1.1知识概述

5.1.1.1 概述

从字面上理解就是别名的意思，和试图的功能类似。就是一种映射关系。

Oracle数据库中提供了同义词管理的功能。同义词是数据库方案对象的一个别名，经常用于简化对象访问和提高对象访问的安全性。在使用同义词时，Oracle数据库将它翻译成对应方案对象的名字。与视图类似，同义词并不占用实际存储空间，只有在数据字典中保存了同义词的定义。在Oracle数据库中的大部分数据库对象，如表、视图、同义词、序列、存储过程、包等等，数据库管理员都可以根据实际情况为他们定义同义词。

5.1.1.2 语法

create synonym 名字 for 表/视图/序列;

同义词的类型：共有和私有；如果使用了public关键字则为公有同义词，任何用户都可以访问，否则只有同义词的创建之才能访问。

Object包含表、视图、序列

5.2 创建同义词

5.2.1知识概述

私有同义词：

create sysnonym 名字 for 表/视图/序列;

共有同义词：

create public sysnonym 名字 for 表/视图/序列;

5.3 索引-概述

5.3.1知识概述

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低i/o 次数,从而提高数据访问性能。

索引是需要占据存储空间的，也可以理解为是一种特殊的数据。形式类似于下图的一棵“树”，而树的节点存储的就是每条记录的物理地址，也就是我们提到的伪列（ROWID）。

5.4 索引-普通索引

5.4.1 知识概述

语法：

create index 名称 on 表名(列名)

性能测试：

```
--创建表
create table T_INDEXTEST (
  ID NUMBER,
  NAME VARCHAR2(30)
);
--使用plsql代码向表中插入1000000条记录
--无索引时插入39.422s/41.438s
--有索引时插入47.156s/42.359s
BEGIN
  FOR i in 1..1000000
  loop
    INSERT INTO T_INDEXTEST VALUES(i,'AA'||i);
  end loop;
  commit;
END;
--创建索引
CREATE INDEX INDEX_TESTINDEX on T_INDEXTEST(name);
SELECT * from T_INDEXTEST where ID=765432;
SELECT * from T_INDEXTEST where NAME='AA765432';
```

上图可以明显看出name查询占用资源比id查询要少很多。
查询的时间也可以看出来使用索引列查询效率高。
5.4.3 总结与补充
　不适合创建index的情况：表很小，数据经常被修改，列很少作为查询条件

第六节
6.1 索引-唯一索引和复合索引
6.1.1 知识概述
6.1.1.1 唯一索引
概述

（1）唯一索引意味着不会有两行记录相同的索引键值。唯一索引表中的记录没有RowID，不能再对其建立其他索引。在oracle10g中，要建立唯一索引，必须在表中设置主关键字，建立了唯一索引的表只按照该唯一索引结构排序。

语法
create uniqune index 名称 on 表名(列名)
6.1.1.2 复合索引
概述：
基于两个以上的列建立一个索引 。
语法：
create uniqune index 名称 on 表名(列名,列名...)
6.2 索引-反向键索引
6.2.1 知识概述

当载入一些有序数据时，索引肯定会碰到与I/O相关的一些瓶颈。在数据载入期间，某部分索引和磁盘肯定会比其他部分使用频繁得多。为了解决这个问题，可以把索引表空间存放在能够把文件物理分割在多个磁盘上的磁盘体系结构上。
为了解决这个问题，Oracle还提供了一种反转键索引的方法。如果数据以反转键索引存储，这些数据的值就会与原先存储的数值相反。这样，数据1234、1235和1236就被存储成4321、5321和6321。结果就是索引会为每次新插入的行更新不同的索引块。

技巧：如果您的磁盘容量有限，同时还要执行大量的有序载入，就可以使用反转键索引。
不可以将反转键索引与位图索引或索引组织表结合使用。因为不能对位图索引和索引组织表进行反转键处理。

语法：

create index 名称 on 表名(列名) reverse

6.3 索引-位图索引

6.3.1 知识概述

位图索引非常适合于决策支持系统(Decision Support System，DSS)和数据仓库，它们不应该用于通过事务处理应用程序访问的表。它们可以使用较少到中等基数(不同值的数量)的列访问非常大的表。尽管位图索引最多可达30个列，但通常它们都只用于少量的列。

例如，您的表可能包含一个称为Sex的列，它有两个可能值：男和女。这个基数只为2，如果用户频繁地根据Sex列的值查询该表，这就是位图索引的基列。当一个表内包含了多个位图索引时，您可以体会到位图索引的真正威力。如果有多个可用的位图索引，Oracle就可以合并从每个位图索引得到的结果集，快速删除不必要的数据。

优点：减少响应时间，节省空间占用。

语法

create bitmap index 名称 on 表名(列名)

6.4 今日总结

6.4.1 知识概述

1、视图

2、物化视图

3、序列

4、同义词

5、索引

```
CREATE  METERIALIZED VIEW view_name
[BUILD IMMEDIATE  | BUILD DEFERRED ]
REFRESH [FAST|COMPLETE|FORCE]
[
ON [COMMIT |DEMAND ] | START WITH (start_time) NEXT (next_time)
]
AS
subquery
```

回帖

3条回帖

范大奎

厉害了我的哥

沙发 • 2017-5-21 16:54:05

回帖

hrywxn

好多内容啊

藤椅 • 2017-5-22 22:15:23

回帖

Aliveqin

楼主最好人

板凳 • 2017-5-27 23:11:11

回帖