王子星OCP的空间

查看文章

REDO 和UNDO的区别

2009-02-08 09:51

redo--> undo-->datafile
insert一条记录时, 表跟undo的信息都会放进 redo 中, 在commit 或之前, redo 的信息会放进硬盘上. 故障时, redo 便可恢复那些已经commit 了的数据.

redo->每次操作都先记录到redo日志中，当出现实例故障（像断电），导致数据未能更新到数据文件，则数据库重启时须redo，重新把数据更新到数据文件
undo->记录更改前的一份copy，但你系统rollback时，把这份copy重新覆盖到原来的数据

redo->记录所有操作，用于恢复（redo records all the database transaction used for recovery）
undo->记录所有的前印象，用于回滚（undo is used to store uncommited data infor used for rollback）

redo->已递交的事务,实例恢复时要写到数据文件去的

undo->未递交的事务.

redo的原因是：每次commit时，将数据的修改立即写到online redo中，但是并不一定同时将该数据的修改写到数据文件中。因为该数据已经提交，但是只存在联机日志文件中，所以在恢复时需要将数据从联机日志文件中找出来，重新应用一下，使已经更改数据在数据文件中也改过来！

undo的原因是：在oracle正常运行时，为了提高效率，加入用户还没有commit,但是空闲内存不多时，会由DBWR进程将脏块写入到数据文件中，以便腾出宝贵的内存供其它进程使用。这就是需要UNDO的原因。因为还没有发出commit语句，但是oracle的dbwr进程已经将没有提交的数据写到数据文件中去了。

undo 也是也是datafile， 可能dirty buffer 没有写回到磁盘里面去。
只有先redo apply 成功了，才能保证undo datafile 里面的东西都是正确的，然后才能rollback

做undo的目的是使系统恢复到系统崩溃前(关机前)的状态,再进行redo是保证系统的一致性.
不做undo,系统就不会知道之前的状态,redo就无从谈起

所以instance crash recovery 的时候总是先rollforward， 再rollback

undo
回退段中的数据是以“回退条目”方式存储。
回退条目=块信息(在事务中发生改动的块的编号)+在事务提交前存储在块中的数据

在每一个回退段中oracle都为其维护一张“事务表”
在事务表中记录着与该回退段中所有回退条目相关的事务编号（事务SCN&回退条目）

redo
重做记录由一组“变更向量”组成。
每个变更变量中记录了事务对数据库中某个块所做的修改。
当用户提交一条commit语句时，LGWR进程会立刻将一条提交记录写入到重做日志文件中，然后再开始写入与该事务相关的重做信息。

#事务提交成功后，Oracle将为该事备生成一个系统变更码（SCN）。事务的SCN将同时记录在它的提交记录和重做记录中。

commit
提交事务前完成的工作：
·在SGA区的回退缓存中生成该事务的回退条目。在回退条目中保存有该事务所修改的数据的原始版本。
·在SGA区的重做日志缓存中生成该事务的重做记录。重做记录中记载了该事务对数据块所进行的修改，并且还记载了对回退段中的数据块所进行的修改。缓存中的重做记录有可能在事务提交之前就写入硬盘中。
·在SGA区的数据库缓丰中记录了事务对数据库所进行的修改。这些修改也有可能在事务提交之前就写入硬盘中。

提交事务时完成的工作：
·在为该事务指定的回退段中的内部事务表内记录下这个事务已经被提交，并且生成一个惟一的SCN记录在内部事务表中，用于惟一标识这个事务。

·LGWR后进进程将SGA区重做日志缓存中的重做记录写入联机重做日志文件。在写入重做日志的同时还将写入该事务的SCN。
·Oracle服务进程释放事务所使用的所有记录锁与表锁。
·Oracle通知用户事务提交完成。
·Oracle将该事务标记为已完成。

rollback
回退事务完成的工作：

·Oracle通过使用回退段中的回退条目，撤销事务中所有SQL语句对数据库所做的修改。
·Oracle服务进程释放事务所使用的所有锁
·Oracle通知事务回退成功。
·Oracle将该事务标记为已完成

举个例子:
insert into a(id) values(1);(redo)
这条记录是需要回滚的。
回滚的语句是delete from a where id = 1;(undo)

试想想看。如果没有做insert into a(id) values(1);(redo)
那么delete from a where id = 1;(undo)这句话就没有意义了。

现在看下正确的恢复:
先insert into a(id) values(1);(redo)
然后delete from a where id = 1;(undo)
系统就回到了原先的状态，没有这条记录了。

相关文章:

- e1034. Adding Undo and Redo to...
- Emacs 中的 Undo/Redo
- Undo/Redo的实现原理
- undo 与 redo
- AS3实现undo/redo原理分析
- C#中使用单个对象的方法实现Undo...
- Undo Redo
- GEF 笔记六 Undo,Redo操作
- Automating Undo/Redo with .NET...
- 用Java来实现编辑器的Undo Redo...

更多>>

网友评论:

1　网友:Danny　2009-02-09 20:05 | 回复
飞机飞到厦门去了,,

2 匿名网友 2009-02-09 20:26 | 回复

回复Danny：FEIJI

发表评论：

姓 名： 注册 | 登录

网址或邮箱： (选填)

内 容： 插入表情

验证码： 请点击后输入四位验证码，字母不区分大小写

发表评论