choelea

choelea

浏览: 3176 次

性别:

来自: 广州

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

IT民工%

xserver

sxk4429

玲cc

博客分类

- 全部博客 (29)
- 待确定 (1)
- 数据库 (9)
- J2EE (8)
- Java (3)
- Design (0)
- 工具使用 (3)
- oracle (1)
- SSH (2)
- web server (0)
- HTML (0)
- Common (2)

我的相册

乐观锁与悲观锁-结合hibernate的简介

2009-11-25

Oracle 事务隔离级别 outline

关键字: 隔离, 隔离级别

隔离级别（isoation eve）
隔离级别定义了事务与事务之间的隔离程度。
隔离级别与并发性是互为矛盾的：隔离程度越高，数据库的并发性越差；隔离程度越低，数据库的并发性越好。
ANSI/ISO SQ92标准定义了一些数据库操作的隔离级别：

- 未提交读（read uncommitted）
- 提交读（read committed）
- 重复读（repeatabe read）
- 序列化（seriaizabe）

通过一些现象，可以反映出隔离级别的效果。这些现象有：

- 更新丢失（ost update）：当系统允许两个事务同时更新同一数据是，发生更新丢失。
- 脏读（dirty read）：当一个事务读取另一个事务尚未提交的修改时，产生脏读。
- 非重复读（nonrepeatabe read）：同一查询在同一事务中多次进行，由于其他提交事务所做的修改或删除，每次返回不同的结果集，此时发生非重复读。(A transaction rereads data it has previousy read and finds that another committed transaction has modified or deeted the data.)
- 幻像（phantom read）：同一查询在同一事务中多次进行，由于其他提交事务所做的插入操作，每次返回不同的结果集，此时发生幻像读。(A transaction reexecutes a query returning a set of rows that satisfies a search condition and finds that another committed transaction has inserted additiona rows that satisfy the condition.)

下面是隔离级别及其对应的可能出现或不可能出现的现象

	Dirty Read	NonRepeatabe Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	not possible	Possible	Possible
Repeatabe read	not possible	not possible	Possible
Seriaizabe	not possible	not possible	not possible

ORACE的隔离级别
ORACE提供了SQ92标准中的read committed和seriaizabe，同时提供了非SQ92标准的read-ony。
read committed:

- 这是ORACE缺省的事务隔离级别。
- 事务中的每一条语句都遵从语句级的读一致性。
- 保证不会脏读；但可能出现非重复读和幻像。

seriaizabe：（串行执行事务，并发性最小）

- 简单地说，seriaizabe就是使事务看起来象是一个接着一个地顺序地执行。
- 仅仅能看见在本事务开始前由其它事务提交的更改和在本事务中所做的更改。
- 保证不会出现非重复读和幻像。
- Seriaizabe隔离级别提供了read-ony事务所提供的读一致性（事务级的读一致性），同时又允许DM操作。

如果有在seriaizabe事务开始时未提交的事务在seriaizabe事务结束之前修改了seriaizabe事务将要修改的行并进行了提交，则seriaizabe事务不会读到这些变更，因此发生无法序列化访问的错误。（换一种解释方法：只要在seriaizabe事务开始到结束之间有其他事务对seriaizabe事务要修改的东西进行了修改并提交了修改，则发生无法序列化访问的错误。）
If a serializable transaction contains data manipulation language (DML) that attempts to update any resource that may have been updated in a transaction uncommitted at the start of the serializable transaction, （并且修改在后来被提交而没有回滚），then the DML statement fails. 返回的错误是ORA-08177: Cannot serialize access for this transaction。
ORACE在数据块中记录最近对数据行执行修改操作的N个事务的信息，目的是确定本事务开始时，是否存在未提交的事务修改了本事务将要修改的行。具体见英文：
Oracle permits a serializable transaction to modify a data row only if it can determine that prior changes to the row were made by

wolf1

共 1 张

其他分类

- 我的收藏 (3)
- 我的论坛主题贴 (0)
- 我的所有论坛贴 (0)
- 我的精华良好贴 (0)

最近加入圈子

存档

- 2010-01 (2)
- 2009-12 (20)
- 2009-11 (7)
- 更多存档...

最新评论

评论排行榜

- 突破定向思维
- 分布式技术
- Agitar
- Hibernate 实现oracle 自动增长
- PreparedStatement 预编译原理

RSS

Add to Google

订阅到 鲜果

[什么是RSS?]

transactions that had committed when the serializable transaction began.
To make this determination efficiently, Oracle uses control information stored in the data block that indicates which rows in the block contain committed and uncommitted changes. In a sense, the block contains a recent history of transactions that affected each row in the block. The amount of history that is retained is controlled by the INITRANS parameter of CREATE TABLE and ALTER TABLE. Under some circumstances, Oracle may have insufficient history information to determine whether a row has been updated by a "too recent" transaction. This can occur when many transactions concurrently modify the same data block, or do so in a very short period. You can avoid this situation by setting higher values of INITRANS for tables that will experience many transactions updating the same blocks. Doing so will enable Oracle to allocate sufficient storage in each block to record the history of recent transactions that accessed the block.
The INITRANS Parameter: Oracle stores control information in each data block to manage access by concurrent transactions. Therefore, if you set the transaction isolation level to serializable, you must use the ALTER TABLE command to set INITRANS to at least 3. This parameter will cause Oracle to allocate sufficient storage in each block to record the history of recent transactions that accessed the block. Higher values should be used for tables that will undergo many transactions updating the same blocks.
read-ony:

- 遵从事务级的读一致性，仅仅能看见在本事务开始前由其它事务提交的更改。
- 不允许在本事务中进行DM操作。
- read ony是seriaizabe的子集。它们都避免了非重复读和幻像。区别是在read ony中是只读；而在seriaizabe中可以进行DM操作。
- Export with CONSISTENT = Y sets the transaction to read-ony.

read committed和seriaizabe的区别和联系:
事务1先于事务2开始，并保持未提交状态。事务2想要修改正被事务1修改的行。事务2等待。如果事务1回滚，则事务2（不论是read committed还是seriaizabe方式）进行它想要做的修改。如果事务1提交，则当事务2是read committed方式时，进行它想要做的修改；当事务2是seriaizabe方式时，失败并报错"Cannot seriaize access"，因为事务2看不见事务1提交的修改，且事务2想在事务1修改的基础上再做修改。
即seriaizabe不允许存在事务嵌套
具体见英文:
Both read committed and serializable transactions use row-level locking, and both will wait if they try to change a row updated by an uncommitted concurrent transaction. The second transaction that tries to update a given row waits for the other transaction to commit or roll back and release its lock. If that other transaction rolls back, the waiting transaction (regardless of its isolation mode) can proceed to change the previously locked row, as if the other transaction had not existed. However, if the other (blocking) transaction commits and releases its locks, a read committed transaction proceeds with its intended update. A serializable transaction, however, fails with the error "Cannot serialize access", because the other transaction has committed a change that was made since the serializable transaction began.
read committed和seriaizabe可以在ORACE并行服务器中使用。
关于SET TRANSACTION READ WRITE: read write和read committed 应该是一样的。在读方面，它们都避免了脏读，但都无法实现重复读。虽然没有文档说明read write在写方面与read committed一致，但显然它在写的时候会加排他锁以避免更新丢失。在加锁的过程中，如果遇到待锁定资源无法锁定，应该是等待而不是放弃。这与read committed一致。
语句级的读一致性

- ORACE保证语句级的读一致性，即一个语句所处理的数据集是在单一时间点上的数据集，这个时间点是这个语句开始的时间。
- 一个语句看不见在它开始执行后提交的修改。
- 对于DM语句，它看不见由自己所做的修改，即DM语句看见的是它本身开始执行以前存在的数据。

事务级的读一致性

- 事务级的读一致性保证了可重复读，并保证不会出现幻像。

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;

设置增个会话的隔离级别

- ATER SESSION SET ISOLATION_LEVE SERIALIZABLE;
- ATER SESSION SET ISOLATION_LEVE READ COMMITTED;

Choice of Isolation Level

Application designers and developers should choose an isolation level based on application performance and consistency needs as well as application coding requirements.
For environments with many concurrent users rapidly submitting transactions, designers must assess transaction performance

requirements in terms of the expected transaction arrival rate and response time demands. Frequently, for high-performance environments, the choice of isolation levels involves a trade-off between consistency and concurrency.

Read Committed Isolation

For many applications, read committed is the most appropriate isolation level. Read committed isolation can provide considerably more concurrency with a somewhat increased risk of inconsistent results due to phantoms and non-repeatable reads for some transactions.
Many high-performance environments with high transaction arrival rates require more throughput and faster response times than can be achieved with serializable isolation. Other environments that supports users with a very low transaction arrival rate also face very low risk of incorrect results due to phantoms and nonrepeatable reads. Read committed isolation is suitable for both of these environments.
两种情况：（1）在事务量大、高性能的计算环境，需要更高的吞吐量和响应时间；（2）事务数少，并且发生幻影和不可重复读的几率的比较低
Oracle read committed isolation provides transaction set consistency for every query. That is, every query sees data in a consistent state. Therefore, read committed isolation will suffice for many applications that might require a higher degree of isolation if run on other database management systems that do not use multiversion concurrency control.
Read committed isolation mode does not require application logic to trap the "Cannot serialize access" error and loop back to restart a transaction. In most applications, few transactions have a functional need to issue the same query twice, so for many applications protection against phantoms and non-repeatable reads is not important. Therefore many developers choose read committed to avoid the need to write such error checking and retry code in each transaction.

Serializable Isolation

Oracle's serializable isolation is suitable for environments where there is a relatively low chance that two concurrent transactions will modify the same rows and the long-running transactions are primarily read-only. It is most suitable for environments with large databases and short transactions that update only a few rows.
（1）适合于很少存在两个事务同时修改同一条记录的情况
（2）长事务以只读为主
（3）大型数据库并且每个短事务只修改很少的记录
Serializable isolation mode provides somewhat more consistency by protecting against phantoms and nonrepeatable reads and can be important where a read/write transaction executes a query more than once.
Unlike other implementations of serializable isolation, which lock blocks for read as well as write, Oracle provides nonblocking queries and the fine granularity of row-level locking, both of which reduce write/write contention. For applications that experience mostly read/write contention, Oracle serializable isolation can provide significantly more throughput than other systems. Therefore, some applications might be suitable for serializable isolation on Oracle but not on other systems.
All queries in an Oracle serializable transaction see the database as of a single point in time, so this isolation level is suitable where multiple consistent queries must be issued in a read/write transaction. A report-writing application that generates summary data and stores it in the database might use serializable mode because it provides the consistency that a `READ ONLY` transaction provides, but also allows `INSERT`, `UPDATE`, and `DELETE`.

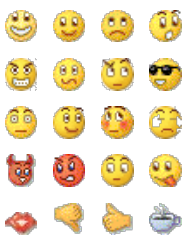
乐观锁与悲观锁-结合hibernate的简介

15:10 | 浏览 (263) | 评论 (0) | 分类: 数据库 | 相关推荐

评论

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字，按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交