On Asymmetric Database Locks

Benet Campderrich

RR85/09

Juliol 85
RESUM

Amb les tècniques usuals de control d'accessos concurrents per mitjà de reserves (locks), el fet que hi incompatibilitat o no entre dues reserves del mateix grànel depèn dels tipus de les reserves però no de l'ordre en què es demanen. Hom estudia la possibilitat de reduir la freqüència dels casos d'incompatibilitat per mitjà de reserves la compatibilitat de les quals pugui dependre de l'ordre en què són demanades.
ON ASYMMETRIC DATABASE LOCKS

Benet Campderrich
Facultat d'Informàtica, Universitat Politècnica de Catalunya
Jordi Girona Salgado, 31
08024 Barcelona, Spain

Abstract

Generally, the occurrence of lock conflicts depends on the types of the locks involved, but not on the order in which the locks are requested. The possibility of reducing the conflict rate through the use of locks not subject to this restriction is studied.

1. Introduction

The concurrency control techniques try to avoid certain anomalies that could happen if simultaneous execution of several transactions is allowed. As far as I know, these anomalies always belong to one of these four classes:

1- A transaction T1 reads granules (records, for example) which have been updated by another transaction T2, which has not committed its updates yet.

2- T1 reads a granule; then T2 updates it; afterwards, T1 does other updates (or processes, in general) which are consistent with the value T1 found, but not with the value T2 has put in the granule.

3- T1 updates granules subject to previous updates from T2, not committed yet.

4- T2 enlarges the write-set or read-set of T1 while T1 is still running.

Some of the most widely used concurrency control techniques use physical locks. Almost always several types of locks are used. If T1 has a lock on granule G and T2 requests a lock on G as well, this second lock can be granted or not depending on the type of each lock. For instance, with locks of the types S ("shared" or
read-only) and X ("exclusive" or update) the compatibility matrix is that of figure 1

\[
\begin{array}{c|c}
T1 & X \\
S & I \\
\hline
S & I \\
X & I \\
\end{array}
\]

Fig. 1

I=Incompatible

which states that the only lock types compatible are S with S.

As far as I know, compatibility matrices are always considered to be symmetric according to one of their diagonals. This means that the compatibility between lock types does not depend on the order in which they are requested by the respective transactions. For example, if T1 has an S-lock on G T2 cannot get an X-lock on G, and if T1 has an X-lock on G T2 cannot get an S-lock on it either. What would happen if the compatibility matrix were not symmetric?

2. Anomalies prevented by symmetric and asymmetric locks

Let us consider first the symmetric case. If both S- and X- locks are held until the transaction commits, class 1, 2 and 3 anomalies are avoided. Class 4 anomalies cannot be prevented in general by physical locks, unless very large granules, like whole files or data bases, are used, because they cannot lock the non-existence of a granule; predicate locking (Eswaran76) or optimistic techniques with logical read-set (Reimer83) must be used instead.

Assume now S- and X- locks with the asymmetric compatibility matrix of figure 2:

\[
\begin{array}{c|c}
T1 & X \\
S' & I \\
\hline
S' & I \\
X & I \\
\end{array}
\]

where T1 is holding a lock when T2 requests one. From this compatibility matrix follows that granules having been read by T1 can be subsequently updated by T2 before T1 finishes, so class 2 anomalies can
occur, while class 1 and 3 anomalies are prevented as in the symmetric case. Class 1 and 3 anomalies will probably always be unacceptable, since if the uncommitted updates from T1 were backed out, either the values read by T2 would have never existed "officially" or updates from T2 would be lost; but class 2 anomalies are not a problem in those transactions whose updates are not dependent on the values of the granules they have read but not updated. In the case of transactions not affected by class 2 anomalies, asymmetric locks allow a higher degree of concurrency than symmetric ones, as S'-X conflicts do not happen.

But some transaction T3 can be subject to class 2 anomalies; with asymmetric locks there are at least two ways to avoid them:

a) T3 requests X-locks on those granules which must not be updated by other transactions before T3 finishes.

b) The locks T3 requests on these granules are of a third type, the S type of the symmetric case; then the compatibility matrix will be like this:

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
<td>X</td>
</tr>
<tr>
<td>S'</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

Solution a) has a big disadvantage: X-S' conflicts can occur between two transactions who try to just read the same granule; solution b) does not introduce unnecessary conflicts, but, like solution a), places on the application programmer the responsibility to decide whether an S- or an S'-lock is required.
3. Asymmetric locks in some special cases

The concept of asymmetric locking can be easily extended to lock techniques which use lock types other than just S and X, for instance locks with several levels of granularity (Gray76) or locks with two versions of the granules (Bayer80).

When several levels of granules are accessed (for example records and pages) three additional lock types are used: IS, IX and SIX. In order to request an X-lock on a record, an IX- or SIX-lock on the page containing the record must be obtained first. Similarly, in order to request an S-lock on a record one must obtain previously an IS, IX, or SIX-lock on the corresponding page. Figures 3 and 4 show respectively the compatibility matrices for the symmetric and asymmetric cases.

![Fig. 3](image1)

![Fig. 4](image2)

With two versions of the granules, a transaction which attempts to update a granule must request an A-type lock on it and is given a private copy of the granule, while the other transactions are still seeing the ancient copy. Two transactions cannot have private copies of the same granule at the same time. On committing, the transaction must request an X-lock on the granule whose copy has updated; if granted, the copy replaces the former version of the granule. The compatibility matrices for the symmetric and asymmetric cases are shown respectively in the figures 5 and 6.

![Fig. 5](image3)

![Fig. 6](image4)
4. Conclusions

Some locking techniques have been presented in which two locks can be compatible or incompatible depending on the order in which they are requested. These techniques may allow a higher degree of concurrency but require a higher involvement of the programmer.

REFERENCEs

Bayer80: Parallelism and recovery in data base systems.
R. Bayer et al.
ACM TODS 5,2 June 1980, p. 139-156

Eswaran76: The notions of consistency and predicate locks in a data base system.
K.P. Eswaran et al.
Comm.ACM 19,11 November 1976, p.624-653

Gray76: Granularity of locks and degrees of consistency in a shared database.
J.N. Gray et al.

Reimer83: Solving the phantom problem by predicate optimistic concurrency control.
M. Reimer