

MaxDB

SQL Locks
Version 7.6

Werner Thesing

THE BEST-RUN BUSINESSES RUN SAP™





Transactions

Locking objects

Locking types

Locking conflict

Configuration

Lock escalations

Monitoring the locking administration

Phenomenons

Isolation Level

Implementation aspects



What is a transaction?

- Sequence of SQL commands
- Data(base) modifications are an atomic unit

Commit  **O.K. accept and fix changes**

Rollback  **Undo changes**

A transaction is a sequence of one or more processing steps. It refers to database objects such as tables, views, joins and so forth.

Here, the following properties must be fulfilled:

■ **Indivisibility**

A transaction is atomic or, in other words, it will either be completely (all of its operations) executed or not at all ("All or nothing principle"). Example: there are no employees without salary.

■ **Consistency**

The defined integrity conditions remain fulfilled. For example, each employee has a personnel number.

■ **Isolation**

The operations within the transaction are isolated from the operations of other transactions.

■ **Permanency**

Changes that transactions have made to objects must be persistent following a system crash, for example.

ACID condition= Atomic, Consistent, Isolation, Durable.



Concurrent (parallel) transactions

- Concurrent access to the same database object



Synchronization logic is required!

If several transactions want to access the same objects concurrently, these accesses must be synchronized with the help of lock management.

Since the database system allows concurrent transactions to access the same database objects, locks are required to isolate individual transactions.

Locking an object means that other transactions are not able to use it in certain ways.

The more locks that are set, and the longer these stay in place, the less concurrency is possible in database operation.

All locks are released by the end of the transaction at the latest.

Locking objects are

- Table rows (ROW)
- Tables (TAB)
- Database catalog (SYS)

Activation

- Implicit
- Explicit

Locking management handles three types of objects:

- Records
- Tables
- Database catalog entries

Requesting locks implicitly

You can choose the lock type by specifying an isolation level when opening the database session. The database system then requests locks implicitly during processing of an SQL statement in accordance with the specified isolation level. All changing SQL statements (such as INSERT, UPDATE, DELETE) always request an exclusive lock.

Requesting locks explicitly

You can use the LOCK statement to explicitly assign locks to a transaction. You can specify a LOCK option in an SQL statement to lock individual rows in a table. This is possible in every isolation level. You can use the LOCK option to temporarily change the isolation level for an SQL statement.

SHARE lock (shared, multiple access)

- Alternate transaction may access the object for reading but not for writing purpose

EXCLUSIVE lock (exclusive access)

- Alternate transactions may access the object for reading pupose but only without a lock (dirty read)

OPTIMISTIC lock

- A transaction (t1) might change an object if and only if no alternate transaction has changed this object after it has been read (by t1, setting the optimistic lock)

Read locks (share locks) refer to a row or a table.

- Once a shared lock is assigned to a transaction for a particular data object, concurrent transactions can access the object but not modify it. Other transactions can set a shared lock, but not an exclusive lock for this object.

Read locks (share locks) refer to a row or a table.

- Once an exclusive lock is assigned a transaction for a particular database object, other transactions cannot modify this object. Transactions that check for the presence of exclusive locks, or that want to set exclusive or shared locks, conflict with the existing exclusive lock of another transaction. You cannot access the locked object.

Optimistic lock on a row level

- An update operation on a row is only actually performed if this row has not been changed in the meantime by a concurrent transaction. If the update operation was successful, an exclusive lock is set for this row. If the update operation was not successful, it should be repeated after reading the row again with or without an optimistic lock. In isolation level 0, an explicit lock must be specified for the new read operation. In this way, it can be ensured that the update is done to the current state and that no modifications made in the meantime are lost.
- It only makes sense to use an optimistic lock if one of the isolation levels 0, 1 or 10, or 15 has been assigned. An optimistic row lock must be explicitly requested by specifying a LOCK statement. A request can conflict with an exclusive lock only.



Can an alternate transaction ... ?	A transaction holds ...					
	EXCL	SHARE	EXCL	SHARE	EXCL	SHARE
	Table lock		Row lock		Catalog lock	
lock this table EXCLUSIVE	NO	NO	NO	NO	NO	YES
lock this table SHARE	NO	YES	NO	YES	NO	YES
lock any row of this table EXCLUSIVE	NO	NO			NO	YES
lock an already locked row EXCLUSIVE			NO	NO		
lock another row EXCLUSIVE			YES	YES		
lock any row of this table SHARE	NO	YES			NO	YES
lock a row SHARE			NO	YES		
lock another row SHARE			YES	YES		
change the table definition in the catalog	NO	NO	NO	NO	NO	NO
read the table definition from the catalog	YES	YES	YES	YES	NO	YES

The above table provides an overview of possible parallel read locks (share locks) and write locks (E locks).

A lock collision exists in the cases which are marked with "No"; i.e., after having requested a lock within a transaction, the user must wait for the lock to be released until one of the above situations or one of the situations that are marked with "Yes" in the matrix occurs.

Additionally, the following applies:

- If no lock has been assigned to a transaction for a data object, then a shared or exclusive lock can be requested within any transaction, and the lock is immediately assigned to the transaction.
- If a shared lock has been assigned to a transaction T for a data object, and if no lock has been assigned to any concurrent transaction for this data object, then the transaction T can request an exclusive lock for this data object and the lock is immediately assigned to this transaction.
- If an exclusive lock has been assigned to a transaction for a data object, then a shared lock can, but need not be requested for this transaction.



Normal

The lock is hold until the end of the transaction. It can, as the case may be, be released explicitly.

Consistent

During a table scan a previously received row lock is released if in return another row of the same table gets locked.

Exclusive until end-of-transaction (eot excl)

A lock has been implicitly set during a write order and for consistency reasons has to be kept until the end of transaction (COMMIT or ROLLBACK).

Temporary

In addition to row locks, a table can be locked SHARE for the duration of a mass command (e.g. update).



DB Kernel Parameters (I)

- | | |
|--------------------------------------|---|
| ■ MAXLOCKS | Max. number of locks |
| ■ MAXUSERTASKS | Max. number of concurrent users |
| ■ LOCK REQUEST TIMEOUT | Max. waiting time for receiving a lock (in seconds) |
| ■ DEADLOCK_DETECTION | Depth level for detecting deadlock cycles |

If a lock request collides with an existing lock:

- the user waits on the existing lock, OR
- an error message is returned for the existing lock.

If the user has to wait (default), he will receive an error message after the lock request has timed out.

Timeouts are updated every 30 seconds by the Timer Task.

A deadlock occurs when two or more users mutually prevent each other from proceeding. Deadlocks are recognized down to a certain depth in the database. The users involved in the deadlock receive an error message. The deadlock is resolved.

Deadlocks that were not recognized by the system are resolved by the timeouts (transactions will be rolled back).

Transfer row locks to a table lock

- if around 20% of the lock list entries ([MAXLOCKS](#)) are used by one single transaction on one table

Reacting to collisions during escalation

- Continue by setting further row locks if other concurrent transactions work on the same table.
- Block execution if the mass command requests more locks than available in lock

A mass command is an SQL statement that affects multiple records.

Example: Update PERSONAL set SALARY (GEHALT) = SALARY* 1.5 where GENDER (GESCHLECHT) = "female" (weiblich)



System tables

- sysdba.lockstatistics
- sysdba.lockliststatistics
- sysdba.transactions

Database console

- x_cons <DBNAME> sh[ow] act[ive]
- the status Vwait shows:
Task is waiting to get an SQL lock

DB50

- SQL lock overview and waiting status

The show command *show stat lock [table]* and *show stat lock config* are no longer supported as of version 7.4. Instead of SHOW commands, SQL statements can be used on system tables.

- SHOW STAT LOCK: SELECT * FROM sysdba.lockstatistics
- SHOW STAT LOCK CONFIG: SELECT * FROM sysdba.lockliststatistics
- SELECT * FROM lock_waits

System Table 'Lockstatistics'



- SESSION internal session id
- TRANSACTION internal transaction id
- PROCESS task id of bound kerneltask
- REQTIMEOUT seconds to return LOCK REQUEST TIMEOUT
- LASTWRITE seconds since last write activity
- LOCKMODE lock entry
- REQMODE lock request entry
- APPLPROCESS process id of the application process (Client)
- APPLNODE computer name (client),
where the application runs on
- OWNER owner of table
- TABLENAME name of table
- ROWIDLENGTH length of locked key
- ROWID locked key
- ROWIDHEX hexadecimal representation of locked key
- ...

© SAP 2007 / MaxDB 7.6 Internals – Locking/Page 12

The system table LOCKSTATISTICS describes the current lock entries and entries for lock requests.

Using the system table LOCKSTATISTICS you can determine the following database information, among other things:

- All locks that are held on a table
- All locks that the current user is holding during his database session (if this is the current user (DBA user) or database system administrator (SYSDBA user), then all locks are displayed).

Users that belong to other user classes only see the locks held by that one user.



Views on sysdba.lockstatistics

- DOMAIN.LOCKS and DOMAIN.LOCK_HOLDER
show all active locks
- DOMAIN.LOCK_REQUESTOR
shows all lock
- DOMAIN.LOCK_WAITS
shows owners of current lock related to current lock requests



SYSDBA.LOCKLISTSTATISTICS

- maximum number of lock entries as defined for locklist
- number of currently used entries
- average number of used entries
- maximum number of used entries
- threshold value for lock escalation
- number of transactions that hold locks
- number of transactions that are requesting locks

You will find a table description of all columns in the system table manual. The following lists only particular columns:

- **MAXLOCKS** contains the number of available locks in the lock list
- **USED ENTRIES** contains the number of entries for locks and lock requests
- **AVG USED ENTRIES** contains the average number of entries used for locks and lock requests
- **MAX USED ENTRIES** contains the maximum number of entries used for locks and lock requests
- **LOCK ESCALATION VALUE** contains the number of table rows from which the lock rows are converted into table locks (lock escalation)
- **LOCK ESCALATIONS** shows the number of escalations occurring so far.
- **LOCK COLLISIONS** shows the number of collisions occurring so far for lock requests.
- **DEADLOCKS** shows the number of deadlocks that have been recognized and resolved by the database system so far.
- **TRANSACTIONS HOLDING LOCKS** contains the number of transactions with assigned locks
- **TRANSACTIONS REQUESTING LOCKS** contains the number of transactions requesting locks

Transaction DB50 Exclusive SQL wait situations



The screenshot shows the SAP DB50 transaction interface. The main window displays 'Exklusive Wartesituationen' with a table of wait situations. The table has the following data:

Task-ID	Appl.-ID	Appl.-Server	Sperrt	Sperrart	Tabellen...	Task-ID	Appl.-ID	Appl.-Server	Wartet	Sperranforderung	War
46	110	10.31.165...	🔒	tab_s...	D010L	33	7058	uw1019	👤	row_exclusive	497

The left sidebar shows a tree view with 'Wartesituationen' selected under 'SQL-Sperren'. The status bar at the bottom indicates 'SQ2 (1) (000) uw1019 INS'.

© SAP 2007 / MaxDB 7.6 Internals – Locking/Page 15

Display of the current wait situations

Task 33 waits for a lock, which can then be assigned only once task 46 has provided the shared table lock.

Exclusive locks prevent other users from accessing the locked entry. These locks can significantly interfere with the performance of the SAP system and the database system.

Procedure to determine the user who triggered the lock

- The column "Appl.ID" displays the process ID of the work process on the application server "Appl.Server". You will find the corresponding SAP work process in transaction SM51/SM50 or SM66.
- The corresponding task (here, task 46) can be aborted in the task manager under "Kernel Threads".

General Overview DB50

Overview SQL Locks



Task-ID	Appl.-ID	Appl.-Server	Sperrart	Sperranforderu...	Status...	Wartezeit auf Sper...	Tabellenname	Zeilen-ID
33	23114	uw1019		row_exclusive	write	5000	D010L	'ZFBAD
33	23114	uw1019	row_exclusive			5000	D010SINF	'ZFBAD
41	1123	uw1019	row_share				SYS%CAT2	x'FF0000
46	110	10.31.165.40	tab_share				D010L	

© SAP 2007 / MaxDB 7.6 Internals – Locking/Page 16

Display of all active and requested database locks.

Exclusive locks prevent other users from accessing the locked entry. These locks can significantly interfere with the performance of the SAP system and the database system.

The system displays detailed information about the locks currently set. This display can be very long in a running SAP system. Therefore, always display the analysis of SQL locks from the overview of wait situations (exclusive).

Task T33 requests a write lock on a record belonging to the table D010L.

Task T46 holds a table lock on table D010L.

Lock Statistics in SQL Studio



SQL Studio [SAPR3,SQ2,uw1019]

File View Direct SQL Window Help

sql SQL QBE (DB) QBE (DDL) ? WAN Mode No Timeout

New SQL Statement

select * from lockstatistics

SESSION	TRANSCOUNT	PROCESS	USERNAME	DATE	TIME	TERMD	EQTIMEOUT	LASTWRITE	LOCKMODE	LOCKSTATE	F
1	1.187	816578	41 SAPR3	09.07.2001	08:30:58	P59954 874	?	?	row_exclusive	?	?
2	1.189	816825	43 SAPR3	09.07.2001	08:31:29	P59954 874	?	?	row_exclusive	?	?
3	1.190	816867	44 SAPR3	09.07.2001	08:31:40	P59954 464	4550	?	?	?	row_ex

Rows in Result: 3 select * from lockstatistics

Auto Commit: On SQL Mode: Internal Isolation Level: Committed Statement successfully executed Execution Time: 08:39:30.082 - 08:39:30.122 (00.040 sec)

New SQL Statement

select * from lockstatistics

REGMODE	REQSTATE	APPLPROCESS	APPLNODE	OWNER	%BLNAY	TABLEID	ROWDLNGTH	ROWMDHEX	ROWID
1 ?	?	2.164	10.18.106.25	SAPR3	ZZTELE	00000000000079C3	69	2041616C6Df	'Aalmink
2 ?	?	2.164	10.18.106.25	SAPR3	ZZTELE	00000000000079C3	69	2041686C66E	'Ahlfeld
3 row_exclusive	?	1.124	10.18.106.25	SAPR3	ZZTELE	00000000000079C3	69	2041616C6Df	'Aalmink

Rows in Result: 3 select * from lockstatistics

Auto Commit: On SQL Mode: Internal Isolation Level: Committed Statement successfully executed Execution Time: 08:40:14.287 - 08:40:14.327 (00.040 sec)

SAPR3 SQ2 uw1019



Phenomena

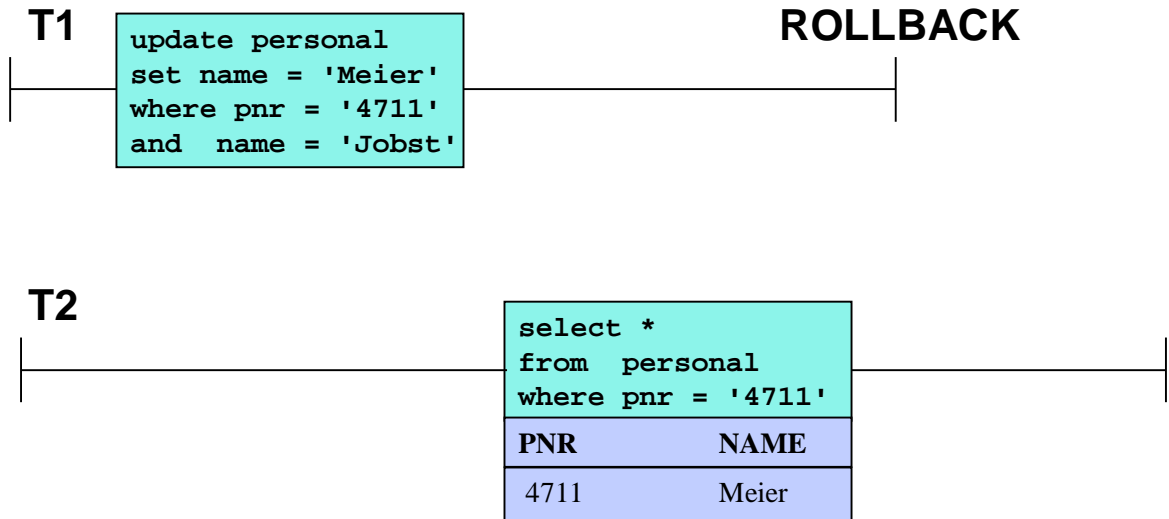
- Dirty Read
- Non Repeatable Read
- Phantom

The isolation level plays an important role in the lock activities of the database system. You use the isolation level to specify whether locks are requested or released implicitly, and how.

Your choice of isolation level affects the degree of parallelism of concurrent transactions and the consistency of the data: the lower the value of the isolation level, the higher the degree of parallelism, and the lower the degree of guaranteed consistency

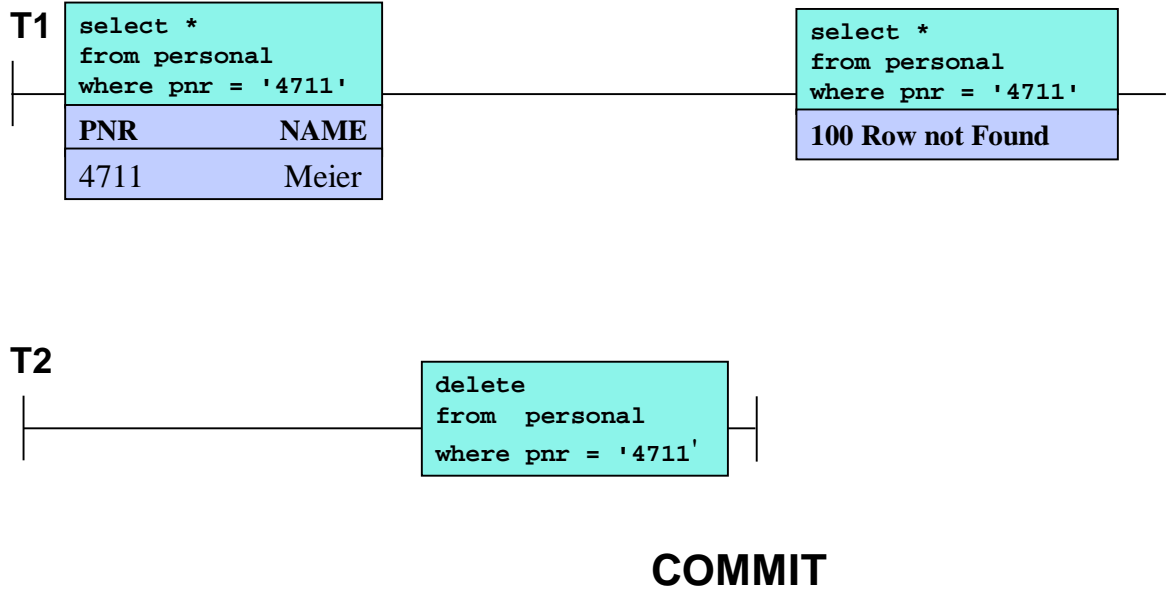
If transactions are competing for access to the same data, then different isolation levels can cause different sorts of inconsistencies. You can find a compromise between parallelism and consistency, while taking into account the requirements of your database application.

When concurrent transactions are processed, inconsistent situations can occur. Try and avoid these situations by configuring the lock behavior and isolation level of the database system accordingly.

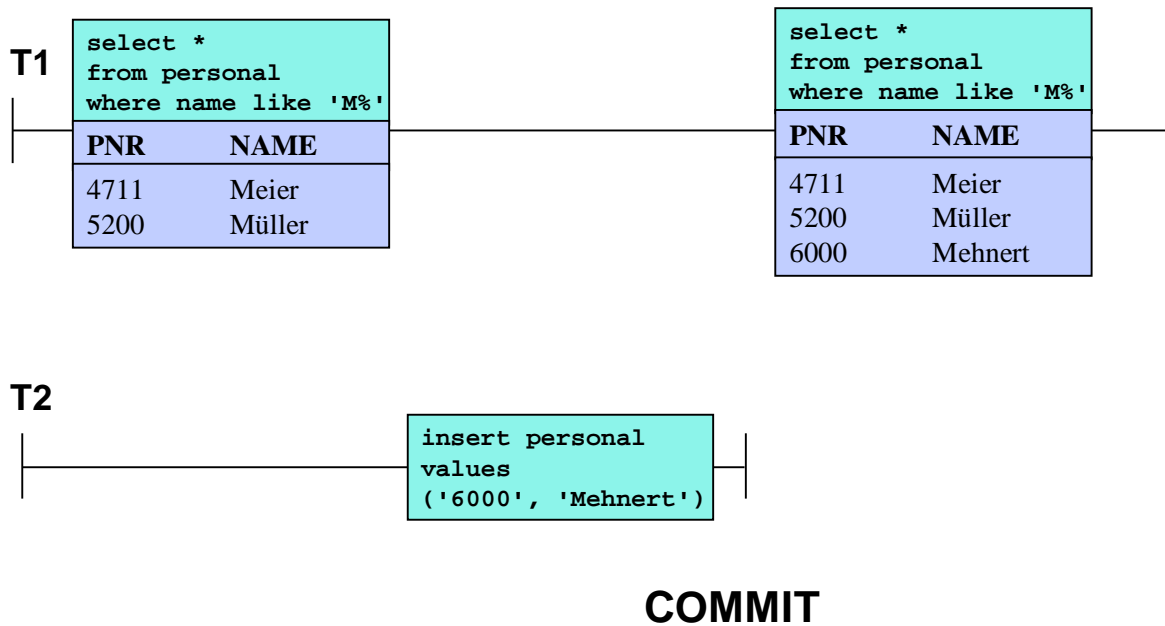


A row is modified in the course of a transaction T1, and a transaction T2 reads this row before T1 has been concluded with the COMMIT statement. T1 then executes the ROLLBACK statements. In this case, T2 read a row that never actually existed.

Non Repeatable Read



Transaction T1 reads a row. Transaction T2 then modifies or deletes this row, and completes the action with the commit statement. If T1 then reads the row again, it either gets the modified row or a message indicating that the row no longer exists.



Transaction T1 executes an SQL statement S that reads a set of rows (M) fulfilling a search condition. Transaction T2 then inserts or modifies data, and produces another row that fulfills this search condition. If T1 then executes the statement S again, the set of rows that is read differs from the set M.



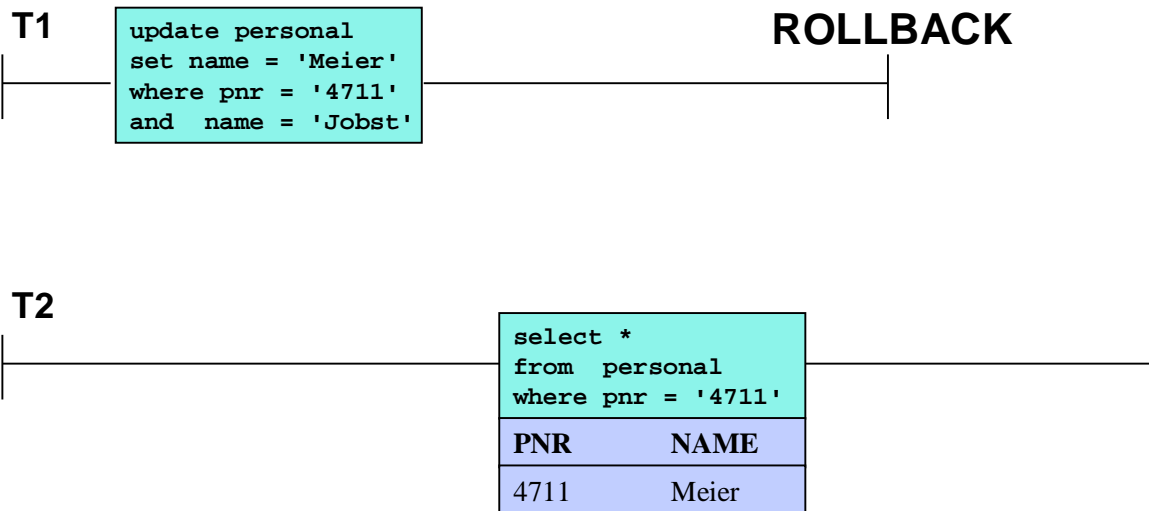
Read access

- Rows are read without checking for lock collisions
- It is not guaranteed that
 - a repeated read within the same transaction returns the same result
 - rows once read ever will be committed (become persistent)

Isolation level 0 does not offer any protection against access anomalies.

If you specify the isolation level 0 (uncommitted), then rows are read without shared locks being requested implicitly. If a row is then read twice within a transaction, this isolation level does not guarantee that the row has the same state the second time as the first, since it could have been changed by a competing transaction between the two reads.

Furthermore, there is no guarantee that the state of a row that was read has already been recorded in the database using a COMMIT WORK statement.

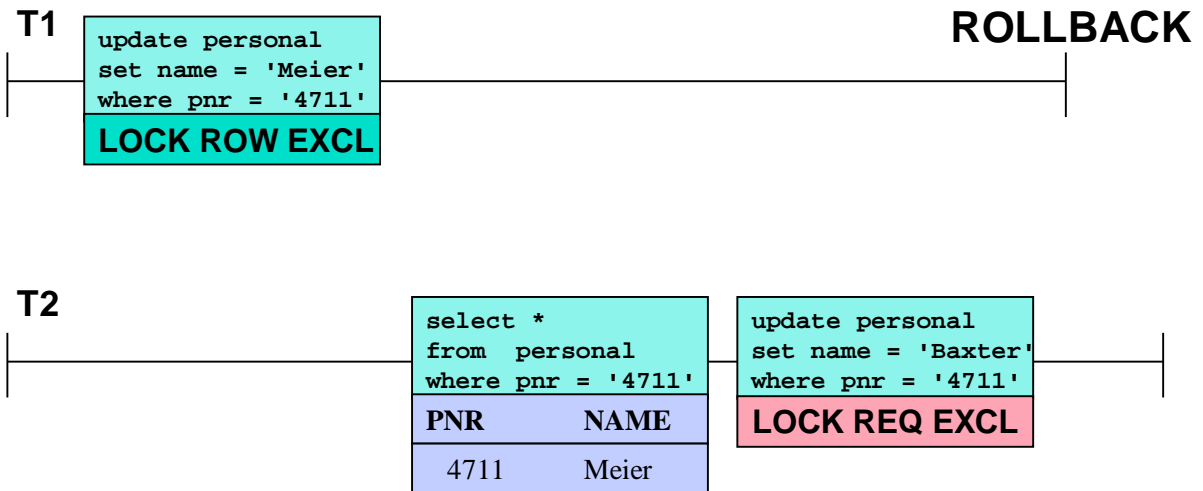




Write access

- Rows are exclusively locked before writing (Insert, Update, Delete)
- Locks are kept until the end of transaction, so that
 - no concurrent modifications can take place
 - this status can be made persistent (committed)

When rows are inserted, updated or deleted, implicit exclusive locks are assigned to the transaction for the rows affected. These cannot be released until the end of the transaction.

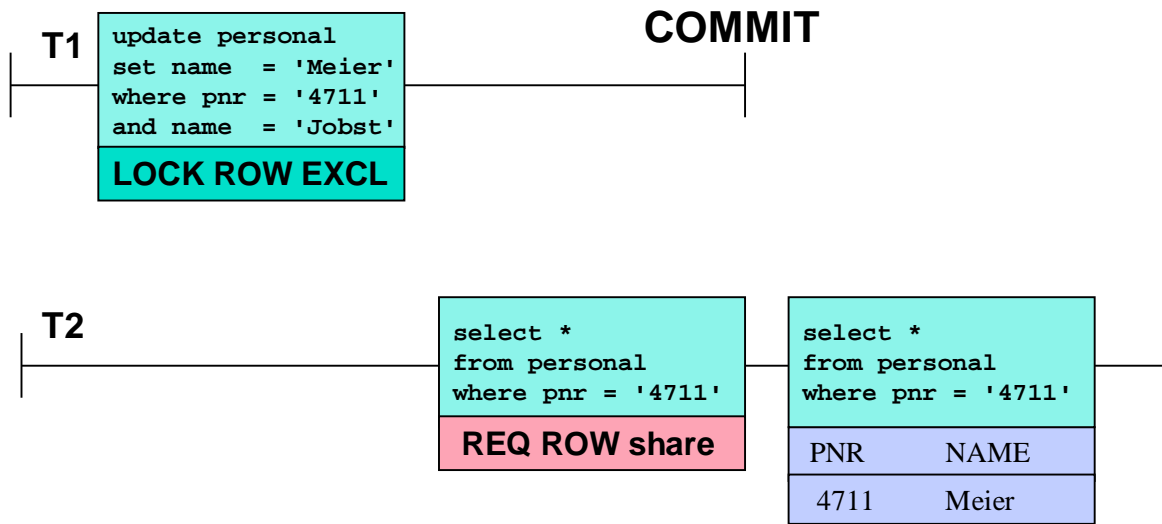


Read Access

- Read persistent (committed) rows. Check for collision happens before reading
- In case of a collision a lock request is set. (req row share)
- The lock dispatcher implicitly changes the request into a lock, if the colliding lock is released. The dispatcher uses a priority list to find the optimum user process.

When you retrieve data using an SQL statement, the database system ensures that, at the time each row is read, no exclusive lock has been assigned to other transactions for the given row. However, it is impossible to predict whether an SQL statement causes a shared lock for a row of the specified table and for which row this may occur. In SAP DB versions < 7.4, the share locks were held until the end of the transaction. In version 7.4 and above, the share lock is removed after the record has been read.

Locking of data entities and optimal multi-user operation are in direct conflict with one another. It is not recognizable whether the waiting user is waiting for a lock or whether the system is running poorly.



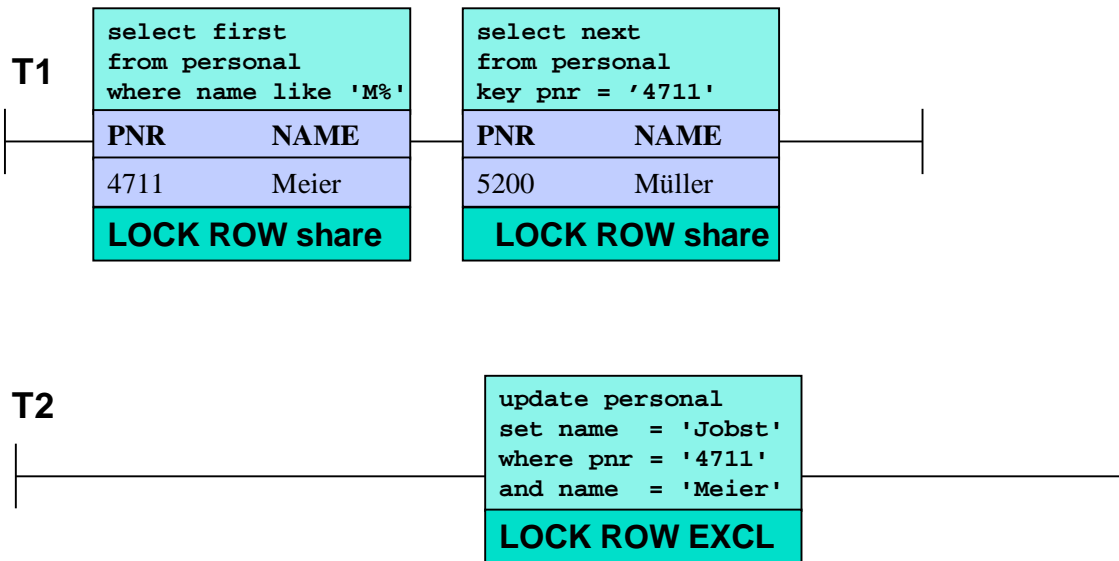


Read Access

- In case of direct key accesses (select direct, select next, select ... key) there will be "moving" share locks

Reading a row will cause setting a share lock for this row simultaneously with releasing the lock from the last row access

If you specify the isolation level 1 or 10 (*committed*), then a shared lock is assigned to the transaction for a read row Z1 of a table. When in the same table the row Z2 is read, the lock on Z1 is released and a shared lock is assigned to the transaction for the row Z2.





Read Access

- Goal:
 - read from tables that are committed for the duration of an SQL command
- Implementation:
 - temporary table locks during execution of the SQL command (tab share)

Isolation level 15 ensures that the result set does not change as long as it is being processed.

Reading backwards and positioning the pointer in the result set generates unique results.

Read Access

- the temporary table lock will be released
 - after execution of the SQL command, if a temporary result table is created (result is copied)
 - after closing of the result table (close)
 - at the end of the transaction

For all SQL statements, the behavior described for isolation level 1 or 10 also applies for isolation level 15: The only difference is that with isolation level 15, shared locks are requested for all the tables addressed by the SQL statement before processing starts. If the SQL statement generates a result table, which is not physically stored, then these locks are not released until the end of the transaction or when the result table is closed. Otherwise, the locks are released immediately after the SQL statement is processed.



T1

```
update personal
set name = 'Lutz'
where pnr = '8500'
and name = 'Bär'
```

REQ ROW EXCL

```
update personal
set name = 'Lutz'
where pnr = '8500'
and name = 'Bär'
```

LOCK ROW EXCL

T2

```
open cursor
select *
from personal
where pnr = '4711'
```

LOCK tab share

```
fetch cursor pos(1)
fetch cursor pos(2)
fetch cursor pos(1)
close cursor
```

PNR	NAME
4711	Meier



T1

```
update personal
set name = 'Lutz'
where pnr = '8500'
and name = 'Bär'
```

LOCK ROW EXCL

T2

```
Open cursor
select *
from personal
where pnr = '4711'
for reuse
```

LOCK tab share

```
fetch cursor
close cursor
```

PNR	NAME
4711	Meier



Write access

- Goal:
 - modify tables that are committed for the duration of an SQL command
- Implementation:
 - temporary table locks during execution of the SQL command (tab share)
 - exclusive row locks (row excl) on new/updated rows until end of transaction

When inserting, changing or deleting rows, the exclusive locks are assigned implicitly to the transaction for the relevant rows that are not released until the end of the transaction.



Read access

- Goal:
 - read from tables that are committed for the duration of an SQL command
 - avoid concurrent follow up modifications to the rows read

- Implementation:
 - temporary table locks during execution of the SQL command (tab share)
 - the rows read are secured from concurrent modifications by using share locks (repeatable read)

Isolation Level 2 safeguards against the "Non Repeatable Read" phenomenon,.
A record that is read multiple times within a transaction always contains the same values.

Read access

- the temporary table lock (tab share) will be released
 - after execution of the SQL command, if a temporary result table is created (result is copied)
 - after closing of the result table (close)
 - at the end of the transaction

- the row locks will be released (row share)
 - at the end of the transaction
 - explicitly with an UNLOCK command

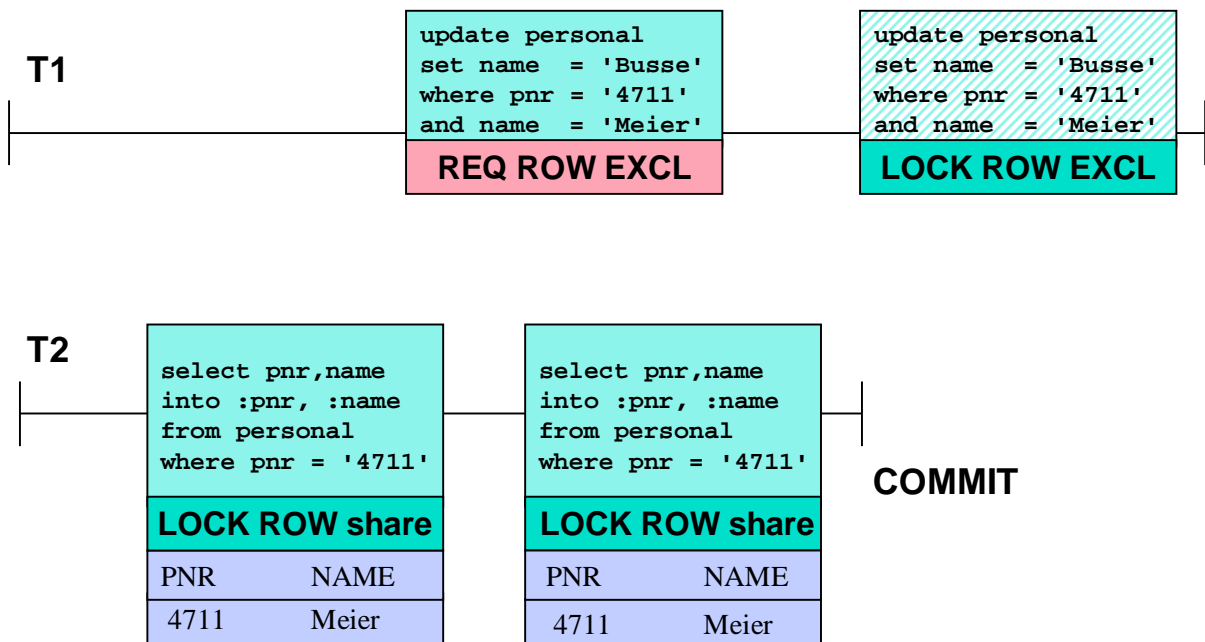
If you specify the isolation level 2 or 20 (repeatable), then shared locks are requested implicitly for all the tables addressed by an SQL statement data query before processing starts.

If an SQL statement generates a result table, which is not physically saved, then these locks are not released until the end of the transaction or when the result table is closed. Otherwise, the locks are released immediately after the SQL statement is processed.

The table shared lock is not assigned to the transaction with SQL statements, where exactly one row in a table is processed that is determined by key specifications or using `CURRENT OF <result_table_name>`.

In addition, an implicit shared lock is assigned to the transaction for each row read while an SQL statement is being processed. These locks can only be released using an UNLOCK statement or by ending the transaction.

ISOLATION LEVEL 2





Write Access

- Goal:
 - modify tables that are committed for the duration of an SQL command

- Implementation:
 - temporary table locks during execution of the SQL command (tab share)
 - exclusive row locks (row excl) on new/updated rows until end of transaction

When inserting, changing or deleting rows, the exclusive locks are assigned implicitly to the transaction for the relevant rows that are not released until the end of the transaction. No locks are set for the whole table, however.



Read access

- Goal:
 - read from tables that are committed for the duration of an SQL command
 - avoid concurrent follow up modifications to the rows read during current transaction (phantom)

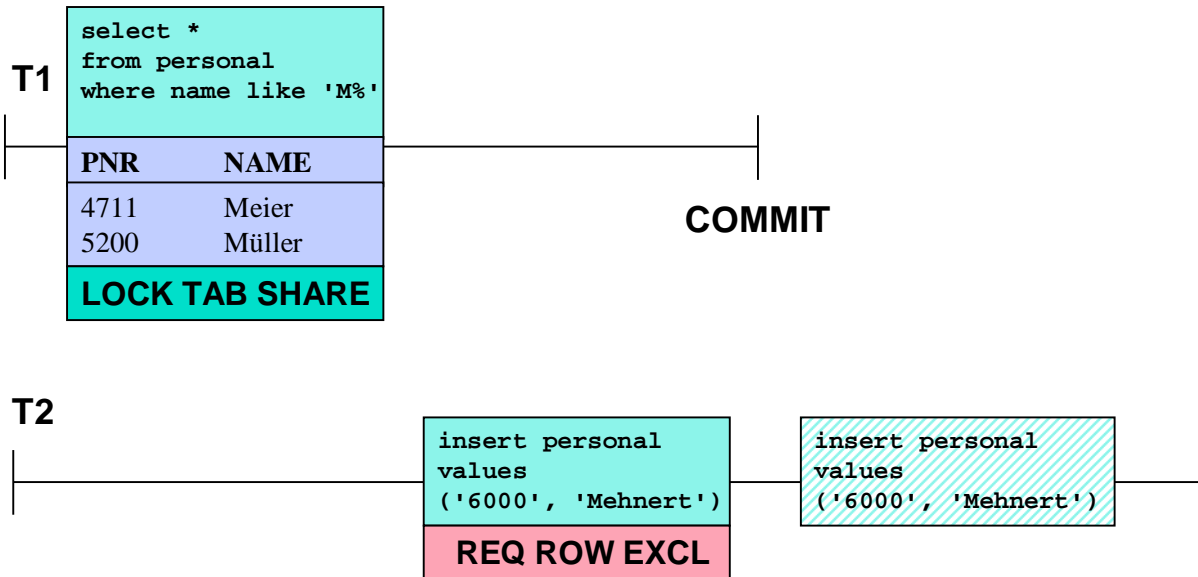
- Implementation:
 - table locks (tab share)
 - release of share locks
 - at the end of the transaction
 - explicitly with an UNLOCK command

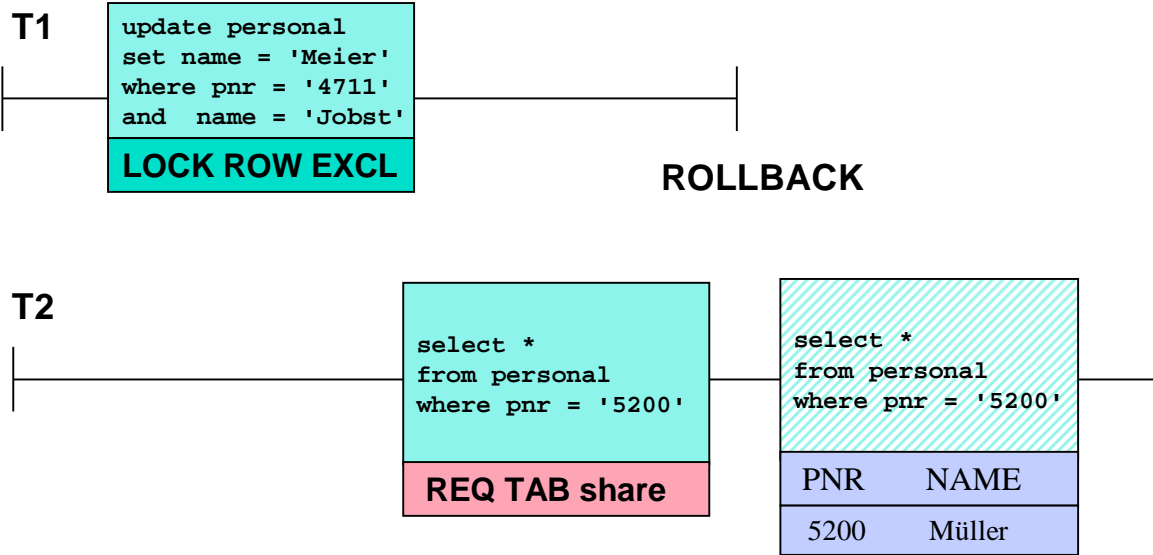
If you specify the isolation level 3 or 30 (serializable), then a table shared lock is implicitly assigned to the transaction for every table addressed by an SQL statement.

These shared locks can only be released by ending the transaction. This table shared lock is not assigned to the transaction with SQL statements, where exactly one row in a table is processed that is determined by key specifications or using `CURRENT OF <result_table_name>`.

Isolation level 3 safeguards against three types of access anomalies:

- Dirty Read
- Non Repeatable Read
- Phantom







Write Access

- Goal:
 - modify tables that are committed for the duration of an SQL command
- Implementation:
 - table locks during execution of the SQL command (tab share)
 - exclusive row locks (row excl) on new/updated rows until end of transaction

When inserting, changing or deleting rows, the exclusive locks are assigned implicitly to the transaction for the relevant rows that are not released until the end of the transaction.



T1

```
update personal
set name = 'Busse'
where pnr = '4711'
and name = 'Meier'
```

MaxDB, DB2, MSSQL, ...

Oracle

T2

```
select *
from personal
where name like 'M%'
```

PNR	NAME
5200	Müller

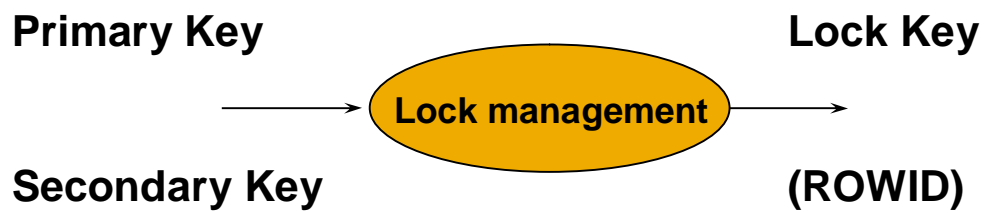
```
select *
from personal
where name like 'M%'
```

PNR	NAME
4711	Meier
5200	Müller



Implementation

- Locking a row (I)
 - a row is locked by blocking the access via primary key
 - if a unique index exists, this secondary key is locked too (secure the one-to-one relation)



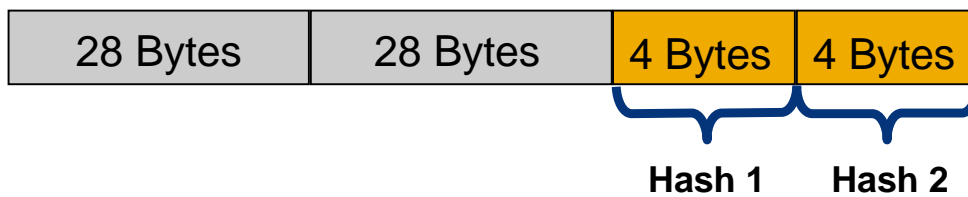
Lock Key Composition



Lock keys have a maximum length of 64 Byte

A lock key consists of 4 parts:

- Start of the primary key (28 Byte)
- Tail of the primary key (28 Byte)
- Generic hash values (2 x 4 Byte)



© SAP 2007 / MaxDB 7.6 Internals – Locking/Page 45

The unique identifier of a record is the primary key. Locks using long primary key values with variable lengths would be inefficient.

Thus, for each primary key a lock key is generated if the primary key is longer than 64 bytes.

As of version 7.4, the lock key has a maximum length of 64 bytes, which is independent of the 64-bit or 32-bit architecture.

The first 56 bytes of the lock key are created from the combination of the first 28 bytes and last 28 bytes of the primary key.

The last 8 bytes of the lock key are generated using two different hash algorithms and with the help of the entire primary key. The result of the first hash algorithm is stored in the second-to-last 4 bytes and the result of the second hash algorithm in the last 4 bytes. Thus, optimum dispersion of the of the lock key values is ensured.



Implementation

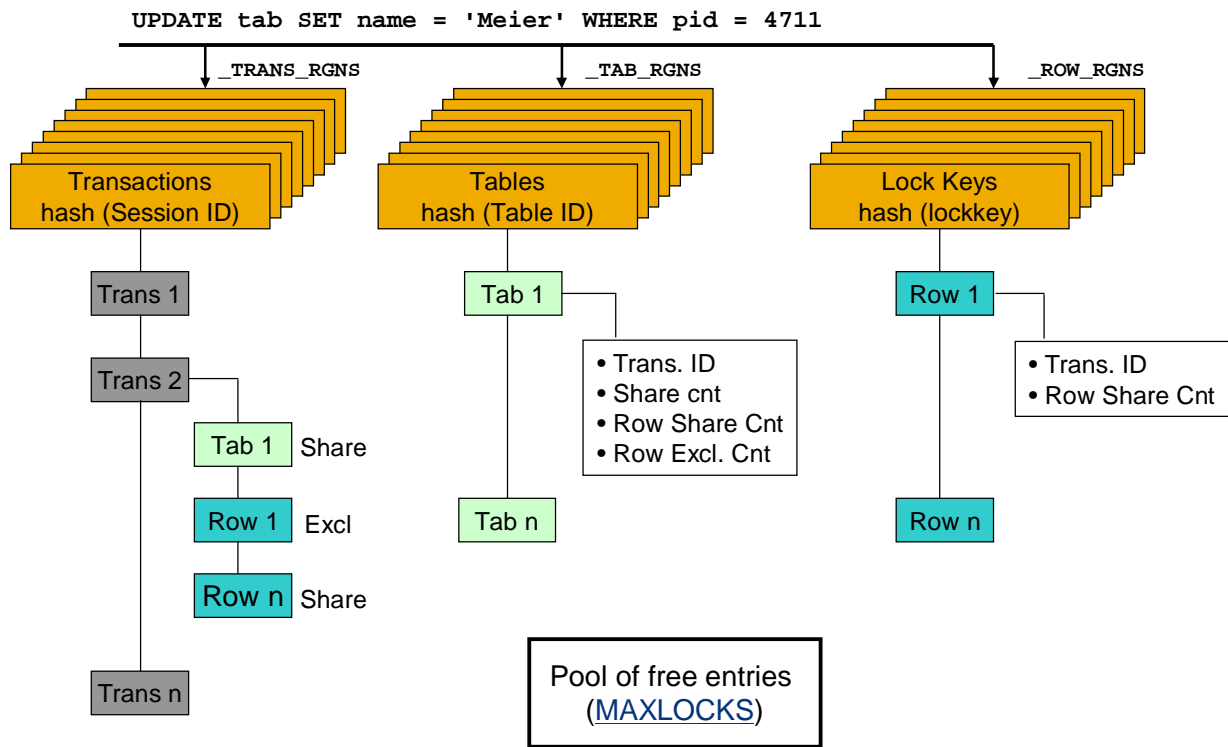
- Locking a row (II)
 - Accesses to LONG values always are secured by locks as reading and writing LONG values can require multiple I/O requests. Thus locking LONGs is independent from the ISOLATION LEVEL)
 - Locks for LONG values are realized as table locks (tab).
 - Each LONG value is identified by a unique TAB-ID, which is used to build the lock entry (i.e. the lock key)

Concept (I)

- Structure:
 - segmental list of transactions
 - segmental list of tables
 - segmental list of lock keys

- Views into the locklist
 - local view on transactions
 - global segment oriented view on tables
 - global segment oriented view on lock keys

Structures of the Locklist



© SAP 2007 / MaxDB 7.6 Internals – Locking/Page 48

Procedure for generating a lock entry:

- Check whether the private transaction view already contains the lock entry (unprotected sequential search).
- If the lock entry is not contained in the private transaction view, the segment assigned to the table of the global table view is checked with respect to a lock collision (hash access).
- In the case of table locks, set the lock or make a lock request
- If the lock entry is not contained in the private transaction view, the segment that is assigned to the table of the global table view is checked with respect to a lock collision (hash access).
- Set the lock or make a lock request for row locks.

The transaction ID displays the exclusive lock in the table and lock key view.

In the table view, Row Share and Exclusive Counter are used to calculate the escalation.

The number of segments of each view is set using the parameter `_TRANS_RGNS`, `_TAB_RGNS` und `_ROW_RGNS`. There are 8 segments, by default.



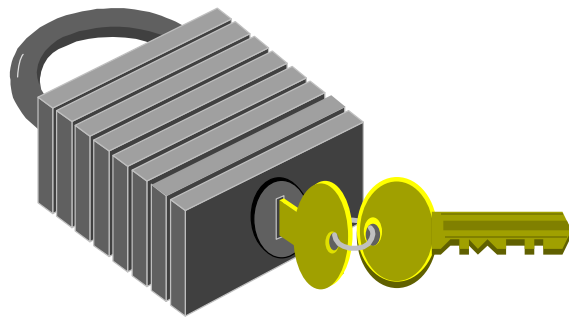
Advantages

- concurrent operation supported by multi-layered segmentation
- hash accesses reduce lookup-time for lock keys

Disadvantages

- memory consumption
- single user operation requires more segment accesses

End



THE BEST-RUN BUSINESSES RUN SAP™





No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, Duet, Business ByDesign, ByDesign, PartnerEdge and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned and associated logos displayed are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.

Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die ausdrückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Einige von der SAP AG und deren Vertriebspartnern vertriebene Softwareprodukte können Softwarekomponenten umfassen, die Eigentum anderer Softwarehersteller sind.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, Duet, Business ByDesign, ByDesign, PartnerEdge und andere in diesem Dokument erwähnte SAP-Produkte und Services sowie die dazugehörigen Logos sind Marken oder eingetragene Marken der SAP AG in Deutschland und in mehreren anderen Ländern weltweit. Alle anderen in diesem Dokument erwähnten Namen von Produkten und Services sowie die damit verbundenen Firmenlogos sind Marken der jeweiligen Unternehmen. Die Angaben im Text sind unverbindlich und dienen lediglich zu Informationszwecken. Produkte können länderspezifische Unterschiede aufweisen.

Die in diesem Dokument enthaltenen Informationen sind Eigentum von SAP. Dieses Dokument ist eine Vorabversion und unterliegt nicht Ihrer Lizenzvereinbarung oder einer anderen Vereinbarung mit SAP. Dieses Dokument enthält nur vorgesehene Strategien, Entwicklungen und Funktionen des SAP®-Produkts und ist für SAP nicht bindend, einen bestimmten Geschäftsweg, eine Produktstrategie bzw. -entwicklung einzuschlagen. SAP übernimmt keine Verantwortung für Fehler oder Auslassungen in diesen Materialien. SAP garantiert nicht die Richtigkeit oder Vollständigkeit der Informationen, Texte, Grafiken, Links oder anderer in diesen Materialien enthaltenen Elemente. Diese Publikation wird ohne jegliche Gewähr, weder ausdrücklich noch stillschweigend, bereitgestellt. Dies gilt u. a., aber nicht ausschließlich, hinsichtlich der Gewährleistung der Marktgängigkeit und der Eignung für einen bestimmten Zweck sowie für die Gewährleistung der Nichtverletzung geltenden Rechts.

SAP übernimmt keine Haftung für Schäden jeglicher Art, einschließlich und ohne Einschränkung für direkte, spezielle, indirekte oder Folgeschäden im Zusammenhang mit der Verwendung dieser Unterlagen. Diese Einschränkung gilt nicht bei Vorsatz oder grober Fahrlässigkeit.

Die gesetzliche Haftung bei Personenschäden oder die Produkthaftung bleibt unberührt. Die Informationen, auf die Sie möglicherweise über die in diesem Material enthaltenen Hotlinks zugreifen, unterliegen nicht dem Einfluss von SAP, und SAP unterstützt nicht die Nutzung von Internetseiten Dritter durch Sie und gibt keinerlei Gewährleistungen oder Zusagen über Internetseiten Dritter ab.

Alle Rechte vorbehalten.