Rdb Journal
technical corner

Guide to Database Maintenance: Locked Free Space Collection Algorithm

A feature of Oracle Rdb

By Ian Smith and Mark Bradley
Oracle Rdb Relational Technology Group
Oracle Corporation

The Rdb Technical Corner is a regular feature of the Oracle Rdb Web Journal. The examples in this article use SQL language from Oracle Rdb V7.1 and later versions.

Guide to Database Maintenance: Locked Free Space Collection Algorithm[1]

This technical note describes the changes that have been made to the locked free space reclamation algorithm in Oracle Rdb version 7.0.

Free space is that part of the data page which is not allocated to stored data (a storage segment), nor used as part of the page structure. This free space shrinks as new rows are stored, and grows when rows are deleted or updated.

Locked free space is that portion of the free space area that must be temporarily reserved for use by executing transactions to guarantee space during recovery (e.g. rollback). This locked free space is eventually (lazily) reclaimed or garbage collected by other transactions over time as those original transactions that locked it are committed.

Space on the page is added to locked free space under the following conditions. Please note that this list is not meant to be exhaustive.

- ERASE operations. Data is erased from a database page when the following actions occur on the database:
    - A DELETE statement erases one or more rows (a side effect of some erase operations may be to also cause an index node to be erased).
    - A TRUNCATE TABLE statement is used to erase table data and the table is in a MIXED format area, or contains LIST OF BYTE VARYING columns.
    - An UPDATE statement assigns a new value (or NULL) to a LIST OF BYTE VARYING column (in this case the old column value is erased).
    - An ALTER TABLE … DROP COLUMN statement removes a LIST OF BYTE VARYING column (all of the now obsolete data is erased at COMMIT time).

[1] This article is a revised version of *Rdb Technical Notes #28, New Locked Free Space Collection Algorithm*

  - DROP TABLE, DROP INDEX, or DROP STORAGE MAP has implicitly erased user data, list segments or index nodes
  - A fragmented row modified by an UPDATE statement could now be stored with fewer fragments. The unused fragments are erased from the pages on which they resided. e.g. a row was fragmented across two pages because originally there was insufficient room on the page, on the next update room now exists and so the fragment on the other page is erased.

- RESIZE operations. Space on the page is added to locked free space when an existing stored segment is reduced in size.
  - The UPDATE statement causes the row image to become smaller. Smaller row images can result from the following:
    - Data that is now more compressible. e.g. columns contain more repeated character sequences (such as trailing spaces) that can now be removed.
    - An ALTER TABLE … DROP COLUMN, ALTER TABLE … ALTER COLUMN, or ALTER DOMAIN statement reduced the size of columns or reduces the number of columns for the table. Subsequent updates now write smaller row images.

    The difference in size is added to locked free space.
  - An INSERT of a LIST OF BYTE VARYING segment recycles a deleted segment as a smaller item of data. The difference in size is added to locked free space.

- STORE failure operations. When a verb failure or ROLLBACK occurs then stored data is also added to locked free space. A verb failure may occur due to a constraint, trigger or other exception. Space used by the storage of a row, index node, duplicate node, hash bucket, etc is returned to **locked** free space.

When data is erased the database page containing that data is modified to indicate that the space containing that data is now free (i.e. is not in use by a table or an index) and that this space is now locked pending the COMMIT of the transaction.

If a verb failure occurs (e.g. a trigger raises the ERROR condition, or uses the SIGNAL SQLSTATE statement, a constraint fails, or some other exception is raised), or if the user executed

a transaction rollback using the ROLLBACK statement the space locked on the page is used by recovery to restore the original data. i.e. the locked/free space guarantees that sufficient space is available to restore the old data even if other rows have been inserted, or existing rows have been expanded due to changes in format or compression.

After the transaction is committed the locked space on the page can be reused by other transactions. This paper examines the improvements made in Rdb V7.0 and later versions in this area of locked free space collection.

Glossary

TID: Although this acronym stands for Transaction Identification number it is really misnamed and represents the attach identification number.

This is a unique value assigned to every database attach and is used in the page structures to identify the database session. However, this TID does not uniquely identify the transactions executed by this database attach.

The TID is a 16-bit value. This is a requirement of the space locking algorithm described below. These values are typically small and are reused frequently by new database attaches.

TSN: This acronym stands for Transaction Sequence Number and is a unique number assigned to every READ WRITE transaction. These numbers start small and continue to increase during the life of the database. Prior to Rdb version 7.0 these values were restricted to an unsigned longword value (i.e. the maximum was approximately 4 billion), however, this limit was lifted in Rdb V7.0 by allowing 64-bit TSN values[2] so that many more transactions can be performed on a database during its lifetime.

DBKEY: When a row is inserted in the database it is assigned a database key (or DBKEY) that describes:

- The storage area (via a logical area number)
- The page within that storage area
- And the line of that page.

[2] To avoid changing the on-disk page layout only 48-bits are used for user data pages. However, this is ample for current needs of Rdb customers.

The line is an offset into two special vectors stored on the page: the TSN index (TDX) which describes the TSN which stored, modified or erased the data, and the LINE index (LDX) which contains an offset to the data within the page, and the length of that data.

When the database attach option **dbkey scope is attach** is used by just one user then all database users will be prevented from reusing entries in these two vectors (the TDX and LDX). These vectors will grow to accommodate new data but will not get reused when those rows are deleted. It is possible that eventually these two indices on the page will consume most of the space on a page. It is for this reason that Oracle suggests not using this mode unless the application absolutely needs to use it.

Rdb also includes an **RMU Reclaim** tool that can be used to release the space on the page that is reserved by the TDX and LDX entries. This tool can be run online and assuming that no database user is attached with DBKEY SCOPE IS ATTACH space on the selected areas will be made available for future updates.

Pad Byte: All storage segments written to a data page will have an even length, Rdb will add a padding byte if needed. This is done so that any change in the record size will leave free multiples of two bytes. The two bytes are required so that the erasing TID (a 16-bit value) can be written into the locked free space section. Please see the algorithms described below.

Minimum Size: All storage segments have a minimum size of 10 bytes, even if the stored size is smaller. This minimum size guarantees that the segment can be fragmented in the case of an update on a page with insufficient free space. Stored table data will have an actual length of at least nine (9) bytes [this includes the record header, row version number, null bit vector and a single minimum column], but data for a LIST OF BYTE VARYING column may require extra pad bytes.

Algorithm Prior to Rdb version 7.0

This space reclamation algorithm was used prior to Rdb V7.0 and is also the only algorithm used by later versions of Rdb if the database was converted from a previous version using **RMU Convert NoCommit**.

- When an ERASE operation is performed
  - The TDX is set to the TSN of the transaction performing the erase
  - The LDX has the offset set to zero (to indicate an erased row) and the length is set to the TID of the erasing database attach

- The locked free space section is expanded to include the space freed by the erased row (which may include the pad byte).

▪ When a RESIZE operation is performed from a larger to a smaller record

- The TDX is set to the TSN of the transaction performing the modify
- The LDX has the length is set to the new smaller size
- The locked free space section is expanded to include the difference between the old and new sizes.

In both cases the TID for the current database attach is written into the area added to the locked free space section. This associates byte pairs with a corresponding TID for future reclamation. Rdb relies on the pad byte to guarantee that locked free space is composed of only byte pairs into which is written the TID.

The SPAM page thresholds assume that locked free space on the page is immediately available. Thus pages with locked free space will be fetched when searching for space to store a new record.

When a page is modified (**insert** or **update**) garbage collection examines the locked free space to see if any can be released. The process confirms that any TID associated with locked free space is no longer active, if so that space is reclaimed for the current updater.

As systems have become faster, and as applications are designed to leave server processes attached for longer periods of time it has become apparent that this scheme has several limitations.

- Firstly, the current erasing process may not reuse this locked free space during this current attachment. This algorithm does not distinguish locked free space for different transactions. Therefore, space is not reused because it may be needed for a rollback operation.
- Secondly, the TIDs may be recycled fairly quickly and so a new process can be active with the same TID as some older attach which erased data. This leads the garbage collection process to incorrectly assume that the erase program is still attached to the database. The end result is that space that could be reclaimed is not, and the storage area is extended.

It is possible that applications generating many verb failures during INSERT are causing locked free space to consume space on the page. Verb failures may be due to failing triggers, constraint violations or unique index checks that cause the rollback of the current statement. This would cause rows, index nodes and duplicate nodes that were just inserted to be erased and added to lock free space.

Note

The UNIQUE index checking logic now avoids creation of a duplicate node and performing node splits prior to reporting the unique index violation. In prior versions this may cause unexpected locked space on the page.

Example of Locked Free Space

The following dumps were made of the PERSONNEL database under Rdb V6.0 to illustrate the locked free space accounting on a page prior to Rdb V7.0.

A SQL query was used to determine the page on which the employee Janet Kilpatrick was stored.

```
SQL> attach 'file DB$:PERSONNEL';
SQL> select first_name, last_name, dbkey
cont> from employees where employee_id='00167';
 FIRST_NAME   LAST_NAME                        DBKEY
 Janet        Kilpatrick                    53:614:3
1 row selected
```

Then the page was dumped and an excerpt included below.

```
$ rmu/dump/area=rdb$system/start=641/end=641 db$:personnel

                0001 00000266  0000  page 614, physical area 1
                     87AC8C46  0006  checksum = 87AC8C46
           00979292 92E571E0  000A  time stamp = 27-JAN-1994 11:46:11.07
                  0000 0064  0012  100 free bytes, 0 locked  (1)
                       000A  0016  10 lines
                  0054 039A  0018  line 0: offset 039A, 84 bytes
                  004E 034C  001C  line 1: offset 034C, 78 bytes
                  0052 02FA  0020  line 2: offset 02FA, 82 bytes
                  0050 02AA  0024  line 3: offset 02AA, 80 bytes  (2)
                  004B 025E  0028  line 4: offset 025E, 75 bytes
                  0048 0216  002C  line 5: offset 0216, 72 bytes
                  0050 01C6  0030  line 6: offset 01C6, 80 bytes
                  0052 0174  0034  line 7: offset 0174, 82 bytes
                  0058 011C  0038  line 8: offset 011C, 88 bytes
                  004F 00CC  003C  line 9: offset 00CC, 79 bytes
```

```
                          00000016  0040  line 0: TSN 22
                          00000016  0044  line 1: TSN 22
                          00000016  0048  line 2: TSN 22
                          00000016  004C  line 3: TSN 22  (3)
                          00000016  0050  line 4: TSN 22
                          00000016  0054  line 5: TSN 22
                          00000016  0058  line 6: TSN 22
                          00000016  005C  line 7: TSN 22
                          00000016  0060  line 8: TSN 22
                          00000016  0064  line 9: TSN 22

00000000000000000000000000000000  0068  free space '................'
                                  ::::  (5 duplicate lines)
                          00000000  00C8  free space '....'  (4)
   .
   .
   .
                              001D  02AA  line 3 (1:614:3) record type 29
                           00 0001  02AC  Control information
                                    ....  75 bytes of static data
69727461706C694B3736313030000110  02AF   data '...00167Kilpatri'  (5)
3431000C208474656E614A0420836B63  02BF   data 'ck. .Janet. ..14'
4D050098208C2E745320656E69502033  02CF   data '3 Pine St.. ...M'
463635343330484E12208D776F6C7261  02DF   data 'arlow. .NH03456F'
          28F0310057C79ED704C000  02EF   data '.À.×.ÇW.1ð('
   .
   .
   .
                          FFFFFFFF  03EE  snap page pointer -1
                          00000016  03F2  snap pointer TSN 22
                              0035  03F6  logical area 53
                          00000000  03F8  page sequence number 0
                          00000000  03FC  MBZ '....'
```

Note the following details:

1. The page header shows that there are initially no bytes of locked free space.
2. The LDX shows the row size of 80 bytes and the offset on the page.
3. The TDX shows the TSN that inserted (or last updated) this row.
4. There is no locked free space marked.
5. From the dump of the record at line 3 you can observe the EMPLOYEE_ID, LAST_NAME and other details of this employee.

Next the Janet Kilpatrick employee row was deleted.

```
SQL> attach 'file DB$:PERSONNEL';
SQL> delete from employees where employee_id='00167';
1 row deleted
SQL> commit;
```

```
                    0001 00000266  0000  page 614, physical area 1
                         C1FEF6DA  0006  checksum = C1FEF6DA
                 009ABA8C B52A9E59  000A  time stamp = 20-NOV-1996 16:59:51.10
                         0050 0064  0012  100 free bytes, 80 locked  (1)
                              000A  0016  10 lines
                         0054 039A  0018  line 0: offset 039A, 84 bytes
                         004E 034C  001C  line 1: offset 034C, 78 bytes
                         0052 02FA  0020  line 2: offset 02FA, 82 bytes
                         0002 0000  0024  line 3: locked by 2  (2)
                         004B 02AE  0028  line 4: offset 02AE, 75 bytes
                         0048 0266  002C  line 5: offset 0266, 72 bytes
                         0050 0216  0030  line 6: offset 0216, 80 bytes
                         0052 01C4  0034  line 7: offset 01C4, 82 bytes
                         0058 016C  0038  line 8: offset 016C, 88 bytes
                         004F 011C  003C  line 9: offset 011C, 79 bytes

                          00000016  0040  line 0: TSN 22
                          00000016  0044  line 1: TSN 22
                          00000016  0048  line 2: TSN 22
                          00000058  004C  line 3: TSN 88  (3)
                          00000016  0050  line 4: TSN 22
                          00000016  0054  line 5: TSN 22
                          00000016  0058  line 6: TSN 22
                          00000016  005C  line 7: TSN 22
                          00000016  0060  line 8: TSN 22
                          00000016  0064  line 9: TSN 22

00020002000200020002000200020002  0068  locked space '................'  (4)
                                  ::::  (4 duplicate lines)
00000000000000000000000000000000  00B8  free space '................'
3731303000010E000001001D00000000  00C8  free space '............0017'
6E6165440320857474656C7472614233  00D8  free space '3Bartlett. .Dean'
20656C7065657453203934314710 2085  00E8  free space '. .G149 Steeple '
12208F796F72540300982088656E614C  00F8  free space 'Lane. ...Troy. .'
004C9114A00400004D3536343330484E  0108  free space 'NH03465M......L.'
                         0020F031  0118  free space '1ð .'
   .
   .
   .
(5)
   .
   .
   .
                          00000001  03EE  snap page pointer 1
                          00000058  03F2  snap pointer TSN 88
```

```
    0035  03F6  logical area 53
00000000  03F8  page sequence number 0
00000000  03FC  MBZ '....'
```

Note the following changes:

1. The page header shows that there are now 80 bytes of locked free space.
2. The LDX has been updated to show that TID 2 locks the row.
3. The TDX is changed to reflect the transaction that erased this row.
4. There is now locked free space marked with the TID of the eraser. Only the first 80 bytes (as indicated by the page header) of the free space are marked in this way.
5. The entire page was not reproduced here, however, the row (line 3) has been deleted.

Note

The contents of the free space is not initialized (to avoid the CPU cost of setting this area to zeros) and this dump shows that when the line 3 was deleted the data was shuffled down and leaving a copy of some data. This can be ignored.

New Lock/Free Space Collection Algorithm

The new locked free space collection algorithm forces the garbage collection to use transaction information, as well as database information used in previous versions. Therefore, it now primarily uses the TSN to determine whether or not space can now be reused.

When an ERASE operation is performed

- the TDX is set to the TSN of the transaction performing the erase
- The LDX has the offset set to the zero (to indicate an erased row) and the length is set to the TID of the erasing database attach.

- The locked free space section is expanded to include the space freed by the erased row (which may include the pad byte).

When a RESIZE operation is performed from a larger to a smaller record

- The TDX is set to the TSN of the transaction performing the modify
- The LDX has the length is set to the new smaller size
- The locked free space section is expanded to include the difference between the old and new sizes.

The TID value is still written into the locked free space section so that those bytes corresponding to this TID can later be reclaimed.

It is possible that an older versions database could be converted to the current Rdb release using **RMU Convert**. If a page is updated which contains information used by the original locked free space algorithm then it must be handled correctly by future versions. Therefore, this algorithm is upward compatible with older versions of Rdb.

Some early versions of Rdb 7.0 set the LDX offset to negative value representing the number of bytes deleted. More recent versions of Rdb 7.0 and Rdb 7.1 do not do this, but rather set it to zero. If the page was updated by these older versions you may see output in RMU Dump that shows these negative lengths.

There are really two questions to answer: When can the line and TSN index (LDX/TDX) entries be reclaimed, and when can the locked space be reclaimed?

Reclaiming Locked Lines

When a page is updated it is processed first by scanning the TDX and LDX vectors on the page to determine which LDX entries can be reclaimed.

Locked lines can never be reclaimed while **dbkey scope is attach**.

If the line is in use (positive LDX offset) then the line is in use and cannot be reclaimed

If the line is not in use (negative or zero LDX offset) then it is a candidate and may be reclaimed.

Lines that are marked as free by having the LDX length and offset set to zero, can always be re-used.

Normally an LDX/TDX entry can be reclaimed if:

- The TSN of the line is older than the oldest active transaction, or
- It was locked by the current attach but in a prior transaction.

When Commit To Journal Optimisation (CTJ) is enabled, the old behavior is used and it is reclaimable if:

- It was locked by the current attach in a previous transaction, or
- The TID is not active.

If snapshots are disabled, it is reclaimable only if the TID is not active.

Once the scan is performed, the LDX/TDX is trimmed back to the highest in use LDX/TDX entry. Lines that were candidates for reclamation that are less than the highest used line number have the LDX offset and length set to zero (free for reuse).

Reclaiming Locked Space

To reclaim locked space on a page, the locked space is scanned to examine the TID that deleted the space.

For locked space that is deleted by the current attach (TID), Rdb doesn't free up the locked space, but does re-use it for storing new rows, or extending existing ones.

If the space was locked by a different attach and that attach is not active, then the space can be reclaimed.

If the attach is active, but its current TSN is not read/write or not the same as any TSN on the page, the space can be reclaimed

Reclaimed space is removed from the locked free space on the page and becomes regular free space.

The main benefits of this new algorithm include:

- Free space locked by a previous transaction for the current attach will always be reused without requiring a new attach. This allows space locked by verb failures and rollback to be immediately reclaimed instead of being reclaimed by a different database attach.
- Other attaches can now reuse space locked by an older transaction in a permanent server[3].
- Excessive page searches are avoided because the SPAM thresholds more accurately reflect the quiescent state of the page. The impact is that less I/O will be required to find space for an INSERT.
- Rdb now records the length of data that was erased in the recovery unit journal (RUJ). Therefore, rollback processing is more efficient when undoing an erase operation.
- This new algorithm is upward compatible with prior versions and will reclaim space as well (normally better) than in prior versions.

Example of Locked Free Space

The following dumps were made of the PERSONNEL database under Rdb V7.0 to illustrate the changes to locked free space accounting on a page.

A SQL query was used to determine the page on which the employee Janet Kilpatrick was stored.

```
SQL> attach 'file DB$:PERSONNEL';
SQL> select first_name, last_name, dbkey
cont> from employees where employee_id='00167';
 FIRST_NAME   LAST_NAME                        DBKEY
 Janet        Kilpatrick                   53:662:3
1 row selected
```

Then the page was dumped and an excerpt included below.

```
$ rmu/dump/area=rdb$system/start=662/end=662 db$:personnel
```

[3] For example, a DELETE server used by an ACMS application or the Replication Option for Rdb.

```
                    0001 00000296  0000  page 662, physical area 1
                         EBB5F27A  0006  checksum = EBB5F27A
                 009A8D83 F8E6DAF8  000A  time stamp = 24-SEP-1996 09:33:56.95
                         0000 0064  0012  100 free bytes, 0 locked  (1)
                              000A  0016  10 lines
                         0054 039A  0018  line 0: offset 039A, 84 bytes
                         004E 034C  001C  line 1: offset 034C, 78 bytes
                         0052 02FA  0020  line 2: offset 02FA, 82 bytes
                         0050 02AA  0024  line 3: offset 02AA, 80 bytes  (2)
                         004B 025E  0028  line 4: offset 025E, 75 bytes
                         0048 0216  002C  line 5: offset 0216, 72 bytes
                         0050 01C6  0030  line 6: offset 01C6, 80 bytes
                         0052 0174  0034  line 7: offset 0174, 82 bytes
                         0058 011C  0038  line 8: offset 011C, 88 bytes
                         004F 00CC  003C  line 9: offset 00CC, 79 bytes

                          00000016  0040  line 0: TSN 22
                          00000016  0044  line 1: TSN 22
                          00000016  0048  line 2: TSN 22
                          00000016  004C  line 3: TSN 22  (3)
                          00000016  0050  line 4: TSN 22
                          00000016  0054  line 5: TSN 22
                          00000016  0058  line 6: TSN 22
                          00000016  005C  line 7: TSN 22
                          00000016  0060  line 8: TSN 22
                          00000016  0064  line 9: TSN 22

00000000000000000000000000000000  0068  free space '................'  (4)
                                  ::::  (5 duplicate lines)
                          00000000  00C8  free space '....'
   .
   .
   .
                              001D  02AA  line 3 (1:662:3) record type 29
                           00 0001  02AC  Control information
                                    ....  75 bytes of static data
69727461706C694B3736313030000110  02AF   data '...00167Kilpatri'  (5)
3431000C208474656E614A0420836B63  02BF   data 'ck. .Janet. ..14'
4D050098208C2E745320656E69502033  02CF   data '3 Pine St.. ...M'
4636353433330484E12208D776F6C7261  02DF   data 'arlow. .NH03456F'
         28F0310057C79ED704C000  02EF   data '.À.×.ÇW.1ð('
   .
   .
   .
                          FFFFFFFF  03EE  snap page pointer -1
                          00000016  03F2  snap pointer TSN 22
                              0035  03F6  logical area 53
                          00000000  03F8  page sequence number 0
                              0000  03FC  page TSN base 0
                              0000  03FE  MBZ '..'
```

Note the following details:

1. The page header shows that there are initially no bytes of locked free space.
2. The LDX shows the row size of 80 bytes and the offset on the page.
3. The TDX shows the TSN that inserted (or last updated) this row.
4. There is no locked free space marked.
5. From the dump of the record at line 3 you can observe the EMPLOYEE_ID, LAST_NAME and other details of this employee.

Next the Janet Kilpatrick employee row was deleted.

```
SQL> attach 'file DB$:PERSONNEL';
SQL> delete from employees where employee_id='00167';
1 row deleted
SQL> commit;
```

```
    0001 00000296  0000  page 662, physical area 1
         15C9F71C  0006  checksum = 15C9F71C
009ABA8E B503F199  000A  time stamp = 20-NOV-1996 17:14:09.84
         0050 0064  0012  100 free bytes, 80 locked  (1)
              000A  0016  10 lines
         0054 039A  0018  line 0: offset 039A, 84 bytes
         004E 034C  001C  line 1: offset 034C, 78 bytes
         0052 02FA  0020  line 2: offset 02FA, 82 bytes
         0002 0000  0024  line 3: locked by 2  (2)
         004B 02AE  0028  line 4: offset 02AE, 75 bytes
         0048 0266  002C  line 5: offset 0266, 72 bytes
         0050 0216  0030  line 6: offset 0216, 80 bytes
         0052 01C4  0034  line 7: offset 01C4, 82 bytes
         0058 016C  0038  line 8: offset 016C, 88 bytes
         004F 011C  003C  line 9: offset 011C, 79 bytes

          00000016  0040  line 0: TSN 22
          00000016  0044  line 1: TSN 22
          00000016  0048  line 2: TSN 22
          000000E0  004C  line 3: TSN 224  (3)
          00000016  0050  line 4: TSN 22
          00000016  0054  line 5: TSN 22
          00000016  0058  line 6: TSN 22
          00000016  005C  line 7: TSN 22
          00000016  0060  line 8: TSN 22
          00000016  0064  line 9: TSN 22
```

```
00020002000200020002000200020002  0068  locked space '................'  (4)
                                  ::::  (4 duplicate lines)
00000000000000000000000000000000  00B8  free space '................'
3731303000010E000001001D00000000  00C8  free space '............0017'
6E6165440320857474656C7472614233  00D8  free space '3Bartlett. .Dean'
20656C7065657453203934314710 2085  00E8  free space '. .G149 Steeple '
12208F796F72540300982088656E614C  00F8  free space 'Lane. ...Troy. .'
004C9114A00400004D3536343330484E  0108  free space 'NH03465M......L.'
                        0020F031  0118  free space '1ð .'
    .
    .
    .
(5)
    .
    .
    .
                        00000001  03EE  snap page pointer 1
                        000000E0  03F2  snap pointer TSN 224
                            0035  03F6  logical area 53
                        00000000  03F8  page sequence number 0
                            0000  03FC  page TSN base 0
                            0000  03FE  MBZ '..'
```

Note the following changes:

1. The page header shows that there are now 80 bytes of locked free space.
2. The LDX has been updated to show that TID 2 locks the row.
3. The TSN is changed to reflect the transaction that erased this row.
4. There is now locked free space marked with the TID of the eraser. Only the first 80 bytes (as indicated by the page header) of the free space are marked in this way.
5. The entire page was not reproduced here, however, the row has been deleted.

Note

The contents of the free space is not initialized (to avoid the CPU cost of setting this area to zeros) and this dump shows that when the line 3 was deleted the data was shuffled down and leaving a copy of some data. This can be ignored.

Oracle Rdb
Guide to Database Maintenance: Locked Free Space Collection Algorithm
May 2003

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software that powers the Internet.