

# SELECT<sup>®</sup> Online

The Journal of the International Oracle Users Group

Select Journal  
Home

Regular Columns

Past Issues

General Information

IOUG Home



## Select Magazine - July 2000

Volume 7, No. 4

### Life After Block Corruption

By Leonardo Y. Orellano

It's Monday morning, you had a flat tire on the way in to work, your coffee is cold and your DBA is waiting in your office. It is not looking like a good day. The DBA either wants to schedule that 10-day tuning course in Jamaica, or there is a problem with one of the databases. He stutters: "We have a block corruption in the production database."

Your first response is "We are dead now. Where did I put that resume?"

---

This article is about life after block corruption. It is possible to recover the database, including data within the corrupt block. We will discuss how to determine to which type of segment the corrupt datafile belongs and what steps to take for its recovery. These methods have been tested under a variety of operating systems including HP/UX, Windows NT, Solaris and Digital Unix. However, all of these methods are operating-system independent.

As with all backup and recovery discussions, we must begin with a proven, tested and fully implemented backup strategy. It is also critical that the DBA is trained in recovery procedures. You wouldn't let a doctor perform open-heart surgery without practice. The same applies to your DBA. The DBA should have a separate database available to practice corruption and recovery on a regular basis, keeping his or her knowledge fresh at all times.

This article is not intended in any respect to replace Oracle Worldwide Support. In fact, the first step in any of these methods is to contact support. Use of unsupported parameters such as `_offline_rollback_segments` and `_corrupted_rollback_segments`, can possibly place the database into an unsupported state.

Let's review some basic scenarios you might encounter:

**Block corruption in an Index segment:** If you have a corruption in an index the easiest thing to do is to drop it and recreate it. If you get this error repeatedly on the same datafile or disk then you should consider moving the datafile to another disk spindle.

**Block corruption in a Rollback segment:** There are three different scenarios that can happen when you have corruption in a rollback segment.

- The actual undo block is corrupted;
- The undo header block is corrupted; or
- The corrupted block belongs to a table, index or cluster.

The recommended way to recover from rollback segment corruption is using a backup and applying the archive logs. If you don't have a backup you have different options to try to recover the database depending of where the corruption exists and which version of Oracle you are running.

The first step is to determine what type of rollback segment corruption you have. Create the following event in the INIT.ORA file.

```
event = "10015 trace name context forever, level 10"
```

In pre-7.3 Oracle releases, you will have to use the unsupported init.ora parameter: `_OFFLINE_ROLLBACK_SEGMENTS`. This is required because all active transactions will need to be rolled back before the database can be opened and a corrupt rollback segment will normally prevent this.

**Example:**

```
ROLLBACK_SEGMENTS(RB01,RB02,RB04)
_OFFLINE_ROLLBACK_SEGMENTS(RB03)
```

Oracles releases 7.3 and above do not require this parameter to open the database.

Oracle will now open the database after the roll forward phase is complete. The active transactions are marked 'dead' and are rolled back by either SMON or by another transaction that wants to access the rows locked by the previous transaction. Oracle does not read the transaction table for the rollback segments.

The 'event = ' parameter will dump the transaction recovery tables to a trace file in your user\_dump\_dest directory. You need to search the trace file for a statement like the following:

```
error recovering tx(12,26) object 54
tx is the transaction id and the 54 is the object id
```

If you do not receive this error, the corruption is in the undo header block or an undo block of the rollback segment. To recover from this type of corruption, you will have to use the unsupported `_CORRUPTED_ROLLBACK_SEGMENTS` init.ora parameter.

**Example:**

```
ROLLBACK_SEGMENTS(RB01,RB02,RB04)
_CORRUPTED_ROLLBACK_SEGMENTS(RB03)
```

You can then drop the corrupted rollback segment and recreate it. Caution is advised when doing this as it will mark uncommitted transactions as committed and you will end up with lost or inconsistent data. To be sure that the database will be in a consistent state you should always rebuild the database.

If you do find an error similar to the one above, you have corruption in a data segment or an index segment, and you can run the following query to determine the specific object that contains the corrupted block.

```
SELECT OWNER,OBJECT_NAME FROM DBA_OBJECTS WHERE OBJECT_ID = '
the id from the trace file'.
```

Now you need to know what block in the object is corrupted and depending on the object type do a corresponding recovery.

### Block Corruption in a Data Segment

In this part of the article we make the assumption that the corrupted block is in a data segment.

Normally when you receive an ORA-1578, you have to view to the alert log file to see the exact error message. Normally it looks something like this:

```
'ORA-1578:Data block corrupted in file <F> block <B>'
```

The first thing you need to know is which datafile is involved and what kind of object is affected. The following queries will give you this information:

```
SELECT NAME FROM V$DATAFILE WHERE FILE# = <F>;
```

where <F> is the file number specified in the message from the alert log file.

The second query is for gathering information about the specific object/segment.

```
SELECT OWNER, SEGMENT_NAME, SEGMENT_TYPE FROM DBA_EXTENTS
WHERE FILE_ID = <F> AND <B> BETWEEN BLOCK_ID AND (BLOCK_ID + BLOCKS -1);
```

At this point, you have gathered enough information about the specific objects that seem to be corrupted.

The next step is to verify that we actually have a problem, by analyzing and validating the internal structure of the object by executing the following statement:

```
ANALYZE TABLE EMPLOYEE VALIDATE STRUCTURE;
```

You should run this command at least twice, if you get the same error message,

ORA-1578, then you have a corrupt block. The data in the corrupt block is most probably lost, that is "PROBABLY" lost, not definitely. I will explain, in the last section, about the options you have to try to recover part or all the rows in that block.

The next step is to extract the good data from the corrupted object. The technique differs a bit depending whether you are running Oracle 7.xx or Oracle 8.xx.

### Oracle 7.xx Approach

At this point, we know both the name of the table, the datafile number and block that is corrupt. The first step is to create a temp table exactly as the original:

```
CREATE TABLE TEMP_EMPLOYEE AS SELECT * FROM EMPLOYEE WHERE 1=2;
```

Next step is to insert all the data except the rows that are in the corrupt block:

```
INSERT INTO TEMP_EMPLOYEE SELECT /* + ROWID(EMPLOYEE) */ * WHERE
ROWID NOT LIKE '0000000<B>.%0000000<F>';
```

Where <B> and <F> are the parameters from the ORA-1578 message in the alert.log, <B> corresponds to the block number and <F> corresponds to the file number.

Next step is to drop the corrupt table and rename the temp table to its original name.

Another technique to get the data out of the corrupt table is by using a unique index (you could potentially do it with a non-unique index but you could lose data).

```
SELECT EMPID, ROWID FROM EMPLOYEE WHERE EMPID > 0 AND ROWID LIKE '0000000A.%0003';
```

**Note:** if the index was on a character column you should use: EMPID > "

The returning rows would be something like this:

EMPID	ROWID
12	0000000A.0002.0003
13	0000000A.0005.0003
.	
.	
54	0000000A.000D.0003

The resulting output shows that the range of rows that are inside the corrupted block have an Empid between 12 and 54. This range of rows can be excluded during the population of the temp table. The next step is to create an exact copy of the original table:

```
CREATE TABLE TEMP_EMPLOYEE AS SELECT * FROM EMPLOYEE
WHERE 1=2;
```

And here we populate the temp table excluding the range of empid between 12 and 54:

```
INSERT INTO TEMP_EMPLOYEE SELECT * FROM EMPLOYEE
WHERE EMPID < 12 AND EMPID > 54;
```

Alternatively, you could create several insert statements if you have multiple corruptions.

Drop the EMPLOYEE table and rename the temp table to the original name.

### Oracle 8.xx Approach

In Oracle 8.xx you have three ways to rebuild a corrupt table. The first one is to use a event in the init.ora that ignores blocks that are marked corrupt.

```
Event = "10231 trace name context forever, level 10"
Event = "10232 trace name context forever, level 10"
```

The first event will ignore blocks that are marked "soft corrupt" when you do full-table scans. The second event will create a Hex dump of the block when a corruption is encountered.

The Next step is to export the table, drop it and recreate it by importing it back.

You should comment this event out when you are done because of performance concerns.

This procedure is not bulletproof because it only works when you have a soft corruption (the sequence is set to 0 in the block).

Before we go forward to the other method, we need to have a basic understanding of the extended (Oracle 8.XX) rowid.

The ROWID in Oracle8 is an 18-digit number with the following format:

```
'OOOOOFFFFBBBBBSSS'
```

OOOOOO is a base 64 encoding of the 32-bit dataobj# (Data object number was introduced in Oracle 8.0 to track versions of the same segment because certain operations can change the version.

FFF is a base 64 encoding of the relative file number.

BBBBBB is a base 64 encoding of the block number.

SSS is the base 64 encoding of the slot (row) number.

One important observation concerns the relative file number. This number is relative to the tablespace in contrast to the absolute file number that is absolute to the whole database.

**Note:** When you get an ORA-1578, the file number argument is the absolute file number.

To get the rowid range so we can bypass the corrupted block we need to use the ROWID\_CREATE function that is the DBMS\_ROWID package. The function definition is:

```
rowid_create(rowid_type IN number, object_number IN number, relative_fno
IN number, block_number IN number, row_number IN number) return rowid;
```

rowid\_type Can either be restricted=0 or extended=1; we use extended for Oracle 8.XX.

Object number data object number, which you get by querying TABPART\$.

Relative\_fno Relative file number, which you can get by querying DBA\_EXTENTS.

Block\_number The block number.

Row\_number The row number in the specific block.

The rowid\_type is extended for Oracle8. You can get the object\_number by using the following query:

```
SELECT OBJ# FROM TABPART$ WHERE FILE# = 10 AND BLOCK# = 34;
```

The output will be something like this:

```
OBJ#
1521
```

The file# and block# are the arguments from the ORA-1578 error message, in this example we do the assumption that the <F> is 10 and <B> is 34.

The relative\_fno can you get from the following query:

```
SELECT SEGMENT_NAME, SEGMENT_TYPE, RELATIVE_FNO FROM DBA_EXTENTS
WHERE FILE_ID = 10 AND 34 BETWEEN BLOCK_ID AND BLOCK_ID + BLOCKS -1;
```

The output will be something like this:

SEGMENT_NAME	SEGMENT_TYPE	RELATIVE_FNO
EMPLOYEE TABLE	10	

Now we have all information needed to use the DBMS\_ROWID package.

We know that the corrupt block is 34 (from the error message in the alert.log), the last row that is not in the corrupted is in block 33 and it starts again with the first row in block 35.

Therefore, we are interested in the rows before the corruption and the rows after the corruption. The last row before the corruption can be found with the following query:

```
Select DBMS_ROWID.ROWID_CREATE(1,1521,10,33,0) from employee;
```

The output will be something like this:

```
DBMS_ROWID.ROWID_CREATE
AAAAh4AAFAAAAABAAA
```

The first row after the corruption is:

```
Select DBMS_ROWID.ROWID_CREATE(1,1521,10,35,0) from employee;
```

The output will be something like this:

```
DBMS_ROWID.ROWID_CREATE
AAAAh4AAFAAAAADAAA
```

Next step will be to create a table:

```
CREATE TABLE TEMP_EMPLOYEE AS SELECT * FROM EMPLOYEE WHERE 1=2;
```

Now insert all the 'good' rows:

```
INSERT INTO TEMP_EMPLOYEE SELECT * FROM EMPLOYEE WHERE ROWID < 'AAAAh4AAFAAAAABAAA';
```

```
INSERT INTO TEMP_EMPLOYEE SELECT * FROM EMPLOYEE WHERE ROWID >= 'AAAAh4AAFAAAAADAAA';
```

As in Oracle 7.xx, we have a pretty good chance to recover part or all the rows in the corrupt block by dumping the block the a file, which will be explained in the last section.

### Alternative ways to extract data from a corrupt data block

In Oracle 8.xx, we can dump a block directly from server manager by using the following command:

```
SVRMGR> Alter system dump datafile 13 block 45
```

By using **Table A**, below, you can translate the hex code to ASCII format and put it in a flat file and sql load it in to the database. This procedure tends to be very time consuming, and what I have done in the past is to create a small C-program to do the translation and put the result in a flat file.

The advantage with this method is that Oracle will not perform a sanity check of the block; it will just create a Hex dump. This is a partial example of the Hex dump:

```
col 0: [ 4] 53 43 4f 54 54          SCOTT
col 1: [ 4] 54 49 47 45 52          TIGER
col 2: [ 4] 54 41 4d 50 41          TAMPA
col 3: [ 4] 46 4c 4f 52 49 44 41FLORIDA
col 4: [ 1] 59                      Y
col 5: *NULL*                       'NULL'
```

If you are unable to create a hex dump, then, most probably, you have hardware problems like the controller or even the spindle.

The second way to create a Hex dump is by first shutting down the database, and using the Unix command dd as below:

```
dd bs=4k if=/usr2/oradata/805/ref1.dbf skip=16 count=4 | od -x >
/usr1/dumps
```

where bs=block size, if=input file, skip= skip the first 16 blocks, count= amount of Unix blocks to copy.

You should first consult with the man pages for your Unix flavor or talk with your System Administrator.

The last available resort is Oracle World Wide Support. They use BBED (Block Browser/Editor) which is a block editor where you can edit the footer and the header of a block and patch it. This procedure is very risky and is used normally as a last resort. This editor is not available for customers for obvious reasons.

## Conclusion

Once again, I want to emphasize the importance of a good backup/recovery strategy and remember that the techniques described in this article are not bulletproof. As I said in the beginning of the article, some of these techniques could put your database in an unsupported state, therefore, you should use them with great caution and with supervision from Worldwide Support.

**Table A – Ascii to Hex conversion**

	20	,	2c	8	38	D	44	P	50	\	5c	h	68	t	74
!	21	—	2d	9	39	E	45	Q	51	]	5d	i	69	u	75
“	22	.	2e	:	3a	F	46	R	52	^	5e	j	6a	v	76
#	23	/	2f	;	3b	G	47	S	53	_	5f	k	6b	w	77
\$	24	0	30	<	3c	H	48	T	54	`	60	l	6c	x	78
%	25	1	31	=	3d	I	49	U	55	a	61	m	6d	y	79
&	26	2	32	>	3e	J	4a	V	56	b	62	n	6e	z	7a
‘	27	3	33	?	3f	K	4b	W	57	c	63	o	6f	{	7b
(	28	4	34	@	40	L	4c	X	58	d	64	p	70		7c
)	29	5	35	A	41	M	4d	Y	59	e	65	q	71	}	7d
*	2a	6	36	B	42	N	4e	Z	5a	f	66	r	72	~	7e
+	2b	7	37	C	43	O	4f	[	5b	g	67	s	73	^?	7f

## About the Author

**Leonardo Y Orellano** ([lorella@techdata.com](mailto:lorella@techdata.com)) is a senior Oracle DBA currently working for Tech Data Corporation. He is an Oracle Certified Professional DBA and has more than five years' experience working with the Oracle database and associated products.