Fixing ORA-01452: Identifying and Automating the Removal of Duplicate Keys

Aaron Finley
Certified SAP Netweaver Technology Consultant (Oracle)
www.netweaveradm.com

Fixing ORA-01452: Identifying and Automating the Removal of Duplicate Keys

Overview

ORA-01452 ("cannot CREATE UNIQUE INDEX; duplicate keys found") is a common Oracle error message, especially during `exp`/`imp`/`expdp`/`impdp` tablespace re-organizations. Oracle's DDL definition for an index specifies that the index value sets must be unique per row, however, entries in a table exist which violate this constraint.

This document describes the error messages, problem solving methods, and provides the source code for a tool to automate the mitigation of these errors on import.

Example Scenario: Datapump Import

In this scenario, Oracle datapump utilities (`expdp`/`impdp`) are used to perform an tablespace reorg. Upon import, numerous ORA-01452 entries are found. In order to effectively managed index creation errors, indexes should be created with a SQLFILE. Tables should be loaded without index creation, and then the commands in the SQLFILE should be executed afterwards (preferably with parallelization).

Example `impdp` parfile:

```
directory=expdp_1
dumpfile=expdp_1:expdp_%u.dmp
sqlfile=SQLFILE.SQL
include=INDEX
status=120
job_name=imdp_sql
logfile=impdp_sql.log
```

The above parfile will create an output file (SQLFILE.SQL) containing CREATE INDEX statements.

Example SQLFILE snippet:

```
CREATE UNIQUE INDEX "SAPSR3"."TODIR~0" ON "SAPSR3"."TODIR" ("FRAGID", "FRAGMENT",
"FRAGNAME") PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 377004032 NEXT
1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST
GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "PSAPSR3" PARALLEL 1;
```

To execute the SQLFILE, use SQL*Plus:

```
sqlplus SAPSR3/sap
spool SQLFILE.lst
start SQLFILE
spool off
```

During the processing of the SQLFILE, the following errors are recorded:

```
CREATE UNIQUE INDEX "SAPSR3"."TODIR~0" ON "SAPSR3"."TODIR" ("FRAGID", "FRAGMENT",
"FRAGNAME") PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 377004032 NEXT
1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST
GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "PSAPSR3" PARALLEL 1
*
ERROR at line 1:
ORA-01452: cannot CREATE UNIQUE INDEX; duplicate keys found
```

According to this error message, SAPSR3.TODIR contains duplicate entries for FRAGID, FRAGMENT, and FRAGNAME.

To correct this condition, construct a SQL statement around ROWID:

select *ROWID* from TABLE a WHERE a.*ROWID* > (select MIN(*ROWID*) from TABLE b WHERE b.*KEYS* = a.*KEYS*)

KEYS are the columns specified in the CREATE UNIQUE INDEX statement.

This statement will return the ROWIDs greater than the initial (or minimum) ROWID. This allows the identification of all "duplicate" entries which are present, without the selection of the "original" entry.

For the above example, the SQL statement becomes:

```
SQL> select ROWID from "SAPSR3"."TODIR" a WHERE a.ROWID > (select MIN(ROWID) from
"SAPSR3"."TODIR" b WHERE b.FRAGID = a.FRAGID AND b.FRAGMENT = a.FRAGMENT AND
b.FRAGNAME = a.FRAGNAME);

ROWID
------------------
AAACyhAAkAAAAJfAAj
```

The result returned is one ROWID, meaning that there are two combinations of FRAGID, FRAGMENT, and FRAGNAME which are identical in the table.

To delete the offending ROWID, use the syntax:

DELETE FROM TABLE WHERE *ROWID*='(ROWID');

```
SQL> DELETE FROM "SAPSR3"."TODIR" WHERE ROWID='AAACyhAAkAAAAJfAAj';
SQL> COMMIT;

1 row deleted.
```

Now re-execute the index creation statement:

```
SQL> CREATE UNIQUE INDEX "SAPSR3"."TODIR~0" ON "SAPSR3"."TODIR" ("FRAGID",
"FRAGMENT", "FRAGNAME") PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL
377004032 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS
1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE PSAPSR3 PARALLEL 6;

Index created.
```

After all offending *ROWID*s are deleted, indexes can be recreated simply by re-running the SQLFILE. Existing indexes will not be created. The `ORA-00955: name is already used by an existing object` error can be ignored.

RMDUPKEY.PHP – Automatic SQLFILE spool parser and ROWID identification tool

ORA-01452 errors can affect numerous indexes during an import and the process to identify ROWIDs and correct is manual and tedious.

RMDUPKEY.PHP is a PHP script that performs the following actions automatically:

- Reads the SQLFILE spool output and identifies where errors ORA-01452 have occurred.
- Extracts the table name and columns from index creation statements which failed with ORA-01452.
- Calls SQL*Plus to query the database and returns ROWIDs for duplicate data.
- Writes a SQL deletion script which contains DELETE statements for all relevant ROWIDs.

Syntax:

```
php RMDUPKEY.PHP <SQLFILE.LST> <DELETE_STATEMENTS.SQL>
```

where:

SQLFILE.LST is the spool output generated from running the SQLFILE and DELETE_STATEMENTS.SQL is the location of the file which will contain the relevant ROWID deletion statements.

NOTE: PHP is a highly useful CLI tool in UNIX/Windows environments. For more information, please see http://www.netweaveradm.com/doc.php?id=12

NOTE: The latest version of RMDUPKEY.PHP can be downloaded directly from http://www.netweaveradm.com/doc.php?id=16

Example use:

```
$php rmdupkey.php SQLFILE.LST DELETE_STATEMENTS.SQL

RMDUPKEY.PHP - Remove duplicate table entries.
------------------------------------------------------------------------

Now scanning spool for ORA-01452 errors.................
Now determining table and column names .................
Now determining relevant ROWIDs        .................
Now writing SQL output file            .................
```

Sample DELETE_STATEMENTS.SQL output:

```
SPOOL rmdupkey.lst
DELETE FROM "SAPSR3"."TODIR" WHERE ROWID='AAACyhAAkAAAAJfAAj';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAXAAADWaAB4';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAXAAADXEACr';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAYAAABCVAC/';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAYAAABCnABc';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAXAAADWaAB5';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAXAAADW1AD8';
DELETE FROM "SAPSR3"."THSTC" WHERE ROWID='AAAC2zAAYAAABCnABd';
COMMIT;
```

Execute delete.sql with SQL*Plus to performs deletions. Output will be written to 'rmdupkey.lst'.

RMDUPKEY.PHP Source Code:

```
<?php
// ----------------------------------------------------------------------------
// RMDUPKEY.PHP (Aaron Finley/aaron.finley@atfinley.com)
//
// Function: Parses impdp SQLFILE spool and detects ORA-1452 errors. Generates
//           SQL queries to remove duplicate keys.
//
// Use:      rmdupkey.php <SPOOL_INPUT.LST> <OUTPUT_FILE.SQL>
//
// Notes:    - If your SQLFILE is very large, adjust the memory_limit parameter
//           in php.ini.
//
//           - ORACLE_HOME and ORACLE_SID must be set.
//
//           - SQLPLUS is called via '/ as sysdba'
//
// ----------------------------------------------------------------------------


function sqlplus($SQL) {
  $SQL    = str_replace("\$", "\\$", $SQL);
  $SQL    = str_replace("?","'\"'", $SQL);
  $EXC    = "sqlplus -s /nolog <<EOF\n";
  $EXC   .= "connect / as sysdba\n";
  $EXC   .= "set heading off\nset feedback off\nset echo off\n";
  $EXC   .= $SQL . "\nexit\n";
  $EXC   .= "EOF'";
```

```
  $RAT    = array();
  $RRC    = "";
  exec($EXC, $RAT, $RRC);
  if (array_search("ORA-",$RAT) || $RRC) {
    print   "ERROR: SQL*Plus call failed:";
    print   "\nSQL Statement:\n$SQL\n";
    print   "\nResult:\n";
    print_r ($RAT);
    exit;
  }
  return $RAT;
}

function write_rows_to_sql($array_in,$sqlfile_out) {
  print ("\nNow writing SQL output file            ");
  fwrite($sqlfile_out,"SPOOL rmdupkey.lst\n");
  $array_in_size  = sizeof($array_in);
  for ($n=0;$n<$array_in_size;$n++) {
    $work_table   = $array_in[$n]['name'];
    $work_rowcc   = $array_in[$n]['rows'];
    for ($nn=0;$nn<$work_rowcc;$nn++) {
      $work_rowid = $array_in[$n]["row${nn}"];
      if ($work_rowid) {
        fwrite($sqlfile_out,"DELETE FROM $work_table WHERE " .
                            "ROWID='${work_rowid}';\n");
      }
    }
  print (".");
  }
  fwrite($sqlfile_out,"COMMIT;\nSPOOL OFF\n");
  fclose($sqlfile_out);
}

function determine_rowids($array_in) {
  print("\nNow determining relevant ROWIDs        ");
  $array_in_size  = sizeof($array_in);
  $array_rowids   = array();
  $array_rowids_cc= 0;
  $work_sql       = "";
  for ($n=0;$n<$array_in_size;$n++) {
    $work_table   = $array_in[$n]['name'];
    $work_numcols = $array_in[$n]['cols'];
    $work_sql     = "SELECT ROWID FROM ${work_table} a WHERE a.ROWID > ";
    $work_sql    .= "(SELECT MIN (B.ROWID) FROM ${work_table} b WHERE ";
    for ($nn=0;$nn<$work_numcols;$nn++) {
      $work_col   = $array_in[$n]["col${nn}"];
      $work_sql  .= "b.${work_col} = a.${work_col}";
      if ($nn<($work_numcols-1)) {
        $work_sql.= " AND ";
      }
    }
    $work_sql.=");";
    $sql_result=sqlplus($work_sql);
    $array_rowids[$array_rowids_cc]['name']=$work_table;
    if ($sql_result[0]!="no rows selected") {
      $work_sql_rows = sizeof($sql_result);
      $array_rowids[$array_rowids_cc]['rows']=$work_sql_rows;
      for ($nn=1;$nn<$work_sql_rows;$nn++) {
        $array_rowids[$array_rowids_cc]["row${nn}"]=$sql_result[$nn];
      }
      print (".");
    }
    $work_sql="";
    $array_rowids_cc++;
  }
  return $array_rowids;
}

function extract_table_cols($array_in) {
```

```
  print("\nNow determining table and column names ");
  $array_in_size  = sizeof($array_in);
  $array_index_cols=array();
  $array_index_cols_idx=0;
  $work_col_index=0;
  for ($n=0;$n<$array_in_size;$n++) {
    $work_sql      = $array_in[$n];
    // Determine table name
    $work_explode  = explode('"', $work_sql);
    $table_name    = '"' . $work_explode[5] . '"."' . $work_explode[7] . '"';
    // Determine column names
    $first_lparen  = strpos($work_sql,"(");
    $first_rparen  = strpos($work_sql,")");
    $paren_length  = ($first_rparen-1)-($first_lparen);
    $paren_conten  = substr($work_sql,($first_lparen+1),$paren_length);
    $explode_comm  = explode(",",$paren_conten);
    $explode_commsz= sizeof($explode_comm);
    // Populate array
    $array_index_cols[$array_index_cols_idx]['name']=$table_name;
    $array_index_cols[$array_index_cols_idx]['cols']=$explode_commsz;
    for ($nn=0;$nn<$explode_commsz;$nn++) {
      $work_colid=str_replace('"',"",trim($explode_comm[$nn]));
      $array_index_cols[$array_index_cols_idx]["col${nn}"]=$work_colid;
    }
    $array_index_cols_idx++;
    print (".");
  }
  return $array_index_cols;
}

function find_bad_statements($spool_in,$spool_sz) {
  print ("\nNow scanning spool for ORA-01452 errors");
  $work_buffer_storage="";
  $work_buffer_add=FALSE;
  $work_buffer_hold=FALSE;
  $array_badindex_statements=array();
  $array_badindex_count=0;
  for ($n=0;$n<$spool_sz;$n++) {
    $work_input=strtoupper(trim($spool_in[$n]));
    if (strstr($work_input,"CREATE") && strstr($work_input,"UNIQUE")
        && !strstr($work_input,"ORA-")) {
        $work_buffer_storage="";
        $work_buffer_hold=TRUE;
        $work_buffer_add=TRUE;
    }
    if (strstr($work_input,"*") && $work_buffer_hold==TRUE
        && $work_buffer_add==TRUE) {
      $work_buffer_storage .= " ".$work_input;
      $work_buffer_add=FALSE;
    }
    if (strstr($work_input,"ORA-01452")) {
      $array_badindex_statements[]=$work_buffer_storage;
      $array_badindex_count++;
      print ".";
      $work_buffer_hold=FALSE;
    }
    if ($work_buffer_add==TRUE) {
      $work_buffer_storage .= " ".$work_input;
    }
  }
  return $array_badindex_statements;
}

// Check that ORACLE_HOME and ORACLE_SID are set
if (!isset($_ENV['ORACLE_HOME']) || !isset($_ENV['ORACLE_SID'])) {
  echo "ERROR: ORACLE_HOME and/or ORACLE_SID not set.\n";
  exit;
}
// Check that SPOOL is specified on command line and exists
```

```
if (isset($argv[1]) && file_exists($argv[1])) {
  $spool_in = file($argv[1]);
  $spool_sz = sizeof($spool_in);
}
else {
  echo "ERROR: Could not open SQLFILE.\n";
  exit;
}

if (isset($argv[2])) {
  $sqlfile_out = fopen($argv[2],'w');
  if (!sqlfile_out) {
    echo "ERROR: Could not open output for writing.\n";
    exit;
  }
}
else {
  echo "ERROR: No output file specified.\n";
  exit;
}

print "RMDUPKEY.PHP - Remove duplicate table entries.\n";
print "------------------------------------------------------------------------\n";

$array_of_bad_statements=array();
$array_of_bad_statements=find_bad_statements($spool_in,$spool_sz);
$array_of_index_cols    =extract_table_cols($array_of_bad_statements);
$array_of_row_IDs       =determine_rowids($array_of_index_cols);
write_rows_to_sql($array_of_row_IDs,$sqlfile_out);
print ("\n\n");

?>
```

Additional Information

"Oracle 10/11g Index Compression and Process Automation for SAP Environments" http://www.netweaveradm.com/doc.php?id=20; contains information regarding the automation of index compression in the DataGuard import process.