Oracle 10/11*g* Index Compression and Process Automation for SAP Environments

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

Oracle 10/11*g* Index Compression and Process Automation for SAP Environments

Overview 3

Step by Step Overview of Index Compression 3

Use in a Database reorganization (via DATAPUMP) 5

Appendix: Automation Script for SQLFILE and IX_COMP Parsing 7

Overview

Oracle index compression offers performance and space benefits.

The reference SAP note is 1109743. The note provides the `ind_comp` package which provides a set of utilities for calculating optimal index compression degrees. It generates a SQL script that is used compress indexes. It has a few arguments that control script output, the most important of which are summarized below:

- `#` - number of indexes. Example: 10. Calculate index compression for the 10 largest tables.
- `op_mode` - `ONLINE` or `OFFLINE`. It's a lot faster to schedule application downtime and rebuild indexes in parallel so it is recommended to use `OFFLINE`. `OFFLINE` also works on partitioned and non-partitioned indexes since partitioned indexes must be recreated. Note that this does not shut down Oracle, it means that the script it generates has to be run with SAP shutdown.
- `parallel_degree` - Generated script will build indexes in parallel to this degree (nominal: 6)

The note also ships with a 'typical' OFFLINE and ONLINE script for typical SAP cases. This may be of benefit to a customer supporting a single application with minimal custom developments but it is recommended to run the calculation scripts. The 'typical' scripts will not compress any indexes in the USR tablespace.

Step by Step Overview of Index Compression

1. Create a staging folder for index compression activities. For example, `/oracle/QAS/sapreorg/index_comp`
2. Download `ind_comp3p.txt` from SAP note 1109743 and rename to `ind_comp.sql`. Stage in the index compression folder.
3. Identify schema user, as `ORASID`:

Method #1:

```
env | grep dbs_ora_schema

SAPSR3
```

Method #2:

sqlplus / as sysdba

```
SQL> select OWNER from DBA_TABLES WHERE TABLE_NAME='DDLOG';
```

```
OWNER
-----------------------------
SAPSR3
```

For the purposes of this example 'SAPR3' will be used. If SAP user password is unknown, use `BRTOOLS` to change (menu: 8->4). Note that if system is a dual stack to check/change password for Java database user in `CONFIGTOOL`.

4. Use SQL*Plus to create "~IND_COMP_DIR" directory

```
sqlplus / as sysdba
SQL> CREATE DIRECTORY "~IND_COMP_DIR" as '/oracle/QAS/sapreorg/index_comp ';
SQL> GRANT READ,WRITE ON DIRECTORY "~IND_COMP_DIR" to SAPSR3;
```

5. Load ind_comp PL/SQL package

```
SQL> start ind_comp.sql
```

4. Generate index compression script. The following example will generate index compression scripts for the fifteen largest tables, will include 'parallel 6' for each index build/rebuild command, and will need to be run with the application shutdown (op_mode => 'OFFLINE'). This is the recommended use case.

```
SQL> exec ind_comp.get_column(15, parallel_degree => 6, op_mode => 'OFFLINE');
```

5. The package generates a number of scripts. Most important of the scripts generated is IX_COMP.sql which contains the code to generate all relevant indexes. The script might look as follows:

```
alter index "SAPSR3"."AABLG~0" rebuild  compress 1 parallel 6 pctfree 1;
alter index "SAPSR3"."AABLG~0" noparallel;
--------------------------------------
alter index "SAPSR3"."ACCTIT~0" rebuild  compress 4 parallel 6 pctfree 1;
alter index "SAPSR3"."ACCTIT~0" noparallel;
--------------------------------------
alter index "SAPSR3"."ACCTIT~1" rebuild  compress 4 parallel 6 pctfree 1;
alter index "SAPSR3"."ACCTIT~1" noparallel;
```

6. Some indexes are generated with a very large initial extent. To prevent this, add storage (initial 1m). Using `VI` for example, edit IX_COMP.sql and do '%s/parallel 6/parallel 6 storage (initial 1m)/g'

7. Perform the actual index compression:

Make sure to have good backups before and after. You may want to disable archiving during the rebuild process.

If archiving is disabled during the rebuild/compression process, you must take a backup when it is re-enabled.

If indexes calculated using OP_MODE => 'OFFLINE' then you must shutdown SAP and anything else which accesses the database.

To run IX_COMP:

```
SQL> start IX_COMP
```

Use in a Database reorganization (via DATAPUMP)

Index compression has excellent utility for use with DataPump export/import, which is general is the fastest way to 'reorg' an Oracle database. Index compression can also be performed with standard exp/imp but the process will be much slower.

A standard DataPump import recreates indexes without parallelization. It recommended to perform the import in three parts in order to allow parallelization of index creation:

1. Generate SQLFILE containing DDL statements to create indexes. Sub activities are the creation of the parfile to generate the SQLFILE and the parsing, editing, and cleanup of the SQLFILE
2. Importing table data via 'impdp' without index creation.
3. Creating indexes with SQL*Plus and the edited SQLFILE

Since the SQLFILE contains CREATE INDEX statements it is highly efficient to calculate and include index compression in this process.

Steps:

1. Generate IX_COMP.sql in the source system before export. In the case of production copies, an similar or identical system can be used to generate the DDL.

2. Generate SQLFILE for impdp via parfile, example:

```
directory=expdp_1
dumpfile=expdp_1:expdp_%u.dmp,expdp_2:expdp_2_%u.dmp,expdp_3:expdp_3_%u.dmp
full=y
sqlfile=SQLFILE.SQL
status=120
job_name=impdp_sql
logfile=impdp_sql.log
```

3. Load tables without indexes using impdp via parfile (EXCLUDE=INDEX), example:

```
directory=expdp_1
dumpfile=expdp_1:expdp_%u.dmp,expdp_2:expdp_2_%u.dmp,expdp_3:expdp_3_%u.dmp
content=all
full=y
exclude=INDEX
parallel=16
status=120
job_name=impdp
logfile=impdp.log
```

4. Remove all DDL from SQLFILE except for create index commands. Add parallel to all create index statements. Apply compression as per IX_COMP output (manually with a text order or automate with grep/awk/perl, or use the script in the APPENDIX).

Example SQLFILE statement:

```
CREATE UNIQUE INDEX "SAPSR3"."BALDAT~0" ON "SAPSR3"."BALDAT" ("MANDANT", "RELID",
"LOG_HANDLE", "BLOCK", "SRTF2")
  PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 16384 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "PSAPSR3" PARALLEL 1 ;
```

Example IX_COMP output:

```
alter index  "SAPSR3" . "BALDAT~0"  rebuild compress 4  tablespace PSAPSR3
parallel 6 pctfree 1;
```

Example modified SQLFILE statement:

```
CREATE UNIQUE INDEX "SAPSR3"."BALDAT~0" ON "SAPSR3"."BALDAT" ("MANDANT", "RELID",
"LOG_HANDLE", "BLOCK", "SRTF2")
  PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 16384 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "PSAPSR3" PARALLEL 1 COMPRESS 4;
```

Appendix: Automation Script for SQLFILE and IX_COMP Parsing

IX_APPLY.PHP is a PHP script that performs the following actions:

- Reads the IX_COMP output and generates a listing of indexes to compress and their compression factor values
- Reads the SQLFILE and generates a new SQLFILE with only index creation commands, relevant compression factors, and the parallel parameter specified.
- Converts 'PARALLEL 1' into specified parallel but retains original following ALTER INDEX .. NOPARALELL or PARALLEL *n* syntax without modification.

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 IX_APPLY.PHP can be downloaded directly from http://www.netweaveradm.com/doc.php?id=14

Instructions:

Stage the IX_COMP.SQL and SQLFILE output files. Change the variables as outlined in the source code to point to these files and specify output SQLFILE and log locations. Execute and review log.

```
<?php
// ------------------------------------------------------------------------
// IX_APPLY.PHP (Aaron Finley/aaron.finley@atfinley.com)
//
// Functions: Applies IX_COMP results to SQLFILE. Removes extraneous entries
//            from SQLFILE. Adds PARALLEL to all create index statements
//
// Use:       Modify the following variables to point to
//            correct file locations:
//            $LOCATION_OF_IX_COMP - location of IX_COMP output
//            $LOCATION_OF_SQLFILE - location of SQLFILE
//            $LOCATION_OF_OUTPUT  - output new SQLFILE here
//            $LOCATION_OF_LOG     - output logfile here
//            $PARALELL_DEGREE     - degree to create in parallel
//            to point to file locations and execute
//
// Note:      If your SQLFILE is very large, adjust the memory_limit parameter
//            in php.ini.
//
// ------------------------------------------------------------------------

// Modify the following variables

$LOCATION_OF_IX_COMP    = "IX_COMP.SQL";
$LOCATION_OF_SQLFILE    = "SQLFILE.SQL";
$LOCATION_OF_OUTPUT     = "SQLFILE_WITH_COMPRESSION.SQL";
$LOCATION_OF_LOG        = "SQLFILE_WITH_COMPRESSION.LOG";
$PARALLEL_DEGREE        = "PARALLEL 6";
```

```
// Do not modify below this line

$file_ixcomp    = file($LOCATION_OF_IX_COMP)
                  OR trigger_error("Could not open $LOCATION_OF_IX_COMP",
                                   E_USER_ERROR);
$file_sqlin     = file($LOCATION_OF_SQLFILE)
                  OR trigger_error("Could not open $LOCATION_OF_SQLFILE",
                                   E_USER_ERROR);
$file_sqlout    = fopen($LOCATION_OF_OUTPUT,'w')
                  OR trigger_error("Could not write to $LOCATION_OF_OUTPUT",
                                   E_USER_ERROR);
$file_logout    = fopen($LOCATION_OF_LOG,'w')
                  OR trigger_error("Could not write to $LOCATION_OF_OUTPUT",
                                   E_USER_ERROR);
$file_sqlin_sz  = sizeof($file_sqlin);
$file_ixcomp_sz = sizeof($file_ixcomp);

// Generate array of IX_COMP indexes and compression values

$ix_comp_array = array();

for ($n=0;$n<$file_ixcomp_sz;$n++) {
  $work             = trim($file_ixcomp[$n]);
  $work_compress    = strstr($work,"compress");
  if ($work_compress && strstr($work,"index")) {
    $work_explode   = explode('"', $work);
    $work_name      = '"' . $work_explode[1] . '"."' . $work_explode[3] . '"';
    $work_compfact  = (int)trim(substr($work_compress,8,3));

    fwrite($file_logout, "IX_COMP: Index: [${work_name}] Compress: ".
                         $work_compfact . "\n");

    $ix_comp_array["${work_name}"] = $work_compfact;
  }
}

fwrite($file_logout, "\n". sizeof($ix_comp_array) . "entries in IX_COMP\n");

// Generate new SQLFILE

$work_flag=FALSE;
$work_write=0;
$log_match="";

for ($n=0;$n<$file_sqlin_sz;$n++) {
  $work             = trim($file_sqlin[$n]);
  if (strstr($work,"INDEX") && strstr($work," ON ")) {
    $work_explode   = explode('"', $work);
    $work_name      = '"' . $work_explode[1] . '"."' . $work_explode[3] . '"';
    $work_flag      = TRUE;
    $work_buffer    = "";
  }
  if (strstr($work,"ALTER INDEX")) {
      fwrite ($file_sqlout,$work_buffer."\n");
      fwrite ($file_sqlout,$work."\n");
      $work_write++;
      $work_flag=FALSE;
  }
  if (strstr($work,"PARALLEL") && $work_flag == TRUE) {
    $work = str_replace("PARALLEL 1",$PARALLEL_DEGREE,$work);
  }

  if (strstr($work,";") && $work_flag == TRUE) {
    if (isset($ix_comp_array["${work_name}"]) && !strstr($work,"COMPRESS")) {
      $work_compfact  = $ix_comp_array["${work_name}"];
      $work_newstr    = " COMPRESS ${work_compfact};\n";
      $work_buffer   .= str_replace(";",$work_newstr,$work);
      $log_match     .= "-> $work_name (degree: $work_compfact)\n";
    }
```

```
    else {
      $work_buffer .= " " . $work;
    }
  }
  else {
    if ($work_flag == TRUE) $work_buffer .= " " . $work;
  }
}

fwrite($file_logout,"Parsed $n lines in SQLFILE. Wrote $work_write indexes.\n");
fwrite($file_logout,"\nMatches:\n${log_match}\n");

fclose($file_sqlout);
fclose($file_logout);

?>
```

Example log output:

```
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~0"] Compress: 2
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~AID"] Compress: 1
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~KGR"] Compress: 2
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~KOS"] Compress: 2
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~LID"] Compress: 2
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~MLI"] Compress: 1
IX_COMP: Index: ["SAPSR3"."/SAPSR3/MATLOC~Z01"] Compress: 2

Parsed 5011438 lines in SQLFILE. Wrote 42375 indexes.

-> "SAPSR3"."/SAPSR3/MATLOC~Z01" (degree: 2)
-> "SAPSR3"."/SAPSR3/MATLOC~0" (degree: 2)
-> "SAPSR3"."/SAPSR3/MATLOC~AID" (degree: 1)
-> "SAPSR3"."/SAPSR3/MATLOC~KGR" (degree: 2)
-> "SAPSR3"."/SAPSR3/MATLOC~KOS" (degree: 2)
-> "SAPSR3"."/SAPSR3/MATLOC~LID" (degree: 2)
-> "SAPSR3"."/SAPSR3/MATLOC~MLI" (degree: 1)
```

Example new SQLFILE output snippet for AM_ALERTA0:

```
CREATE INDEX "SAPSR3"."/SAPSR3/AM_ALERTAO" ON "SAPSR3"."/SAPSR3/AM_ALERT"
("MANDT", "PLVER", "AOT", "AT_ID", "OBJECT_ID3", "VFROM", "VTO", "OBJECT_ID1")
PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 16384 NEXT 1048576 MINEXTENTS 1
MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT) TABLESPACE "PSAPSR3" PARALLEL 6 ;
ALTER INDEX "SAPSR3"."/SAPSR3/AM_ALERTAO" PARALLEL 9;
```