



Value Adds



7/28/15

Performance Tuning on SQL & Datastage

Domain/ Service/ Technology /Geo : Telecom

Name of the Project/Support Fn : ETL Team

Name of the Author : Anand M

Date Created : 28-Jul-2015

Value Add – One Click Summary

Performance Tuning on SQL - Index, DBMS Stats, degree and adding suitable Hints on Table : Extraction of huge tables is made easy

Datastage – performance tuning method applied at component level [ORACLE STAGE, JOIN & LOOKUP]

Job execution - through multiple dimensional scripts : Reduce manual efforts involved & save time.

Alert Monitoring - Space issue & currently running jobs

Project level parameter definition - set at project CONFIG file

Description

Project/Context

The business requirement of the project Data Archival is to archive the data of legacy systems Such as Peoplesoft, Singleview, Tallyman, Crazy john, Orange and MVNO_CDR and store the data in RedShift cloud Platform provided by Amazon Web Service for future use.

Purpose

As a Solution to Data Archival Many old years of data from the legacy systems are being extracted.

Processing of huge data certainly needs a defined time to complete the Data transformation and execution of Jobs.

Considering when the source system is a Database, increase the execution speed by query performance tuning technique.

How we did it

Process Adopted

Step by Step description

1. Created index for table if Index doesn't exists.
2. Added hints in the query to improve the performance.
3. Increased the degree of table.
4. Always go for parallel jobs in order to improve execution speed and performance.
5. Scripts were created to monitor the server space and to send alert mails.

Highlights

- Meet the project Crucial delivery deadline.
- Data Validation are possible ahead of the scheduled phase work.
- Faster Execution of Job.
- During the full production execution, Reduced constant monitoring and effective resource planning across 24X7

How we did it

Process Adopted

Step by Step description

Create index for table

- A database index is a data structure that improves the speed of data retrieval operations on a database table.
- Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed.
- Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

How we did it

Process Adopted

Step by Step description

Create index for table

Syntax:

```
CREATE INDEX INDEX_NAME ON TABLE_NAME(COLUMN_NAME);
```

Example:

```
create index FOH_PERSON_IDENT_INX on  
FOH_PERSON_IDENTIFICATION (PERSON_ID,PERSON_IDENT_ID)  
parallel 16;
```

How we did it

Process Adopted

Step by Step description

Change Table Degree & Perform DBMS Statistics

Syntax:

```
ALTER TABLE <table_name> PARALLEL <degree (8,16)>;
```

Example:

```
Alter table FOH_PERSON_IDENTIFICATION parallel 8;
```

DBMS STATS Example:

```
exec DBMS_STATS.GATHER_TABLE_STATS (ownname => 'GTXUSER' ,  
tabname => 'FOH_PERSON_IDENTIFICATION',cascade => true,  
estimate_percent => 100,method_opt=>'for all indexed columns size 1',  
granularity => 'ALL', degree => 8);
```


How we did it

Process Adopted

Step by Step description

Added hints in the query to improve the performance.

- Parallel hints works by creating several parallel query processes that divide the workload of a SQL statement.
- These processes operate on objects in parallel and can therefore greatly reduce overall execution time.

```
SELECT /*+ PARALLEL(RO,16) PARALLEL(ROLN,16) */ to_char(RO.row_added_dttm,'YYYYMM') , COUNT(*)  
FROM SYSADM.PS_RO_HEADER RO,SYSADM.PS_RO_LINE ROLN  
WHERE RO.CAPTURE_ID=ROLN.CAPTURE_ID(+)  
GROUP BY to_char(RO.row_added_dttm,'YYYYMM')  
order by 1
```

How we did it

Process Adopted

Step by Step description

Go for parallel jobs in order to improve execution speed and performance.

- If cardinality of Tables is 1:1 then server job can be used otherwise needs to go with parallel job.
- If it is required to implement Sparse Lookup based on joining Then we can use Server Job else parallel job would be fine.
- If it is required to use some complex stages available in parallel Job then no option we need to stick with Parallel job.



Tallyman_Script

How we did it

Process Adopted

Step by Step description

Scripts were created to monitor the server space and to send alert mails

- Upon reaching the disk space greater than 80% automated mail alert will be send to team.
- This mail alert will help the team to stop the execution of job by giving STOP command to script which triggers the job.
- Thus Alert mails helps to prevent jobs getting Aborted due to Space issue.



Monitoring -
Disk_space

How we did it

Process Adopted

Step by Step description

Scripts were created to monitor the server space and to send alert mails

```
v_reset='cat reset.txt'
while [ "$v_reset" = "RUN" ]
do
DATE=`date`
SUBJECT="EDW Disk Space Usage and Running Jobs as of $DATE "
MAILID="SrikumarK.Vaidyanathan@vodafone.com.au"
COMMAND_1='ps -fu dsadm | grep "phantom DSD.RUN"'
COMMAND_2='df -k /u30 /u14 /u07'
COMMAND_3='df -k /u99 /u40 /u41 /u42 /u43 '
COMMAND_4='ps -fu dsadm | grep "DSD.RUN" | awk '{print $10}' | sort -r'
echo "Dear Team,\n\n\nRUNNING JOBS: \n\n\n${COMMAND_4} \n\n\nRUNNING DS PROCESS: \n\n\n${COMMAND_1} \n\n\nEDW DISK SPACE USAGE STROAGE: \n\n\n${COMMAND_2}\nEDW DISK SPACE USAGE DATASET: \n\n\n${COMMAND_3} \n\n\nRegards, \nDataStage Administrator." | mailx -s "${SUBJECT}" ${MAILID}
echo "Alert mail sent on $DATE"
sleep 1800
v_reset='cat reset.txt'
done

[dsadm edwpstg1: /home/dsadm/eric ]
$ ps -fu dsadm | grep "phantom DSD.RUN"
    dsadm 29561 25816  0 17:48:45 pts/0    0:00 grep phantom DSD.RUN
    dsadm 23307      1  0 14:39:39 ?        0:01 phantom DSD.RUN PSFT_DS_INTERACTIONS_REP_UBS_RECON.2 0/0/1/0/0

[dsadm edwpstg1: /home/dsadm/eric ]
$ df -k /u30 /u14 /u07
/u07
          (/dev/vx/dsk/dgedwdb/vedwdb07) : 123769388 total allocated Kb
          15169704 free allocated Kb
          108599684 used allocated Kb
          87 % allocation used
/u14
          (/dev/vx/dsk/dgedwdb/vedwdb14) : 622270796 total allocated Kb
          87404488 free allocated Kb
          534866308 used allocated Kb
          85 % allocation used
/u30
          (/dev/vx/dsk/dgedwdb/vedwdb30) : 121114782 total allocated Kb
          56630063 free allocated Kb
          64484719 used allocated Kb
          53 % allocation used

[dsadm edwpstg1: /home/dsadm/eric ]
$ df -k /u99 /u40 /u41 /u42 /u43
/u40
          (/dev/vx/dsk/dgedwapp/vedwapp40) : 117397189 total allocated Kb
          110793116 free allocated Kb
          6604073 used allocated Kb
          5 % allocation used
```

Why this is a Value Adds'

Before


- a) Before tuning the query took long time to extract data from table.**
- b) Since the query is taking long time the session also used to get timeout**
- c) During Full Production execution a constant monitoring and resource planning is mandatory across 24X7.**
- d) Threat on schedule slippage, meeting deadlines**

Benefits

- a) Data Validation are possible ahead of the scheduled phase work.**
- b) Table index, degree and hints increased the speed of the query execution and completion of Jobs.**
- c) Job execution through script helped to reduce the manual efforts and time and effort.**
- d) Meet the project crucial delivery.**
- e) Reduce monitoring efforts & flexible resource planning**

Effort & Cost – One Click Summary

Benefit's through Script

Time taken to 1 batch job [with out performance tuning, automated script]	410 mins	 PI
Time taken to 1 batch job [Performance tuning, automated script are in place]	192 mins	
Parallel Execution [8 batches at a time]	Every one hour a new job would be triggered to have effective server bandwidth, Process and server utilization across 24X7	
Number of executions in a month	175	
Effort saved (in hours/month)	3616 hours/ Month [1040 hours]	
Cost saved considering the rate to be 607 AUD for Onsite	16085.53	



Thank You

IT Services
Business Solutions
Consulting