abstract

In this talk, we will tell you everything you need to know about how to do table partitioning in the OpenEDge RDBMS. We look at 4 different examples where we used table partitioning. In each one we provide complete details of exactly how to do it. We cover enabling partitioning, defining, partitions, moving data, checking status, etc.

Armed with this information, you will be ready to try it on your own system at home.

Of course, you will try first on a test box, right? right?
Table Partitioning Case Studies

Gus Bjorklund – DBA

2016 pug challenge americas, manchester, nh, usa
26-29 june 2016
Notices

- Please ask questions as we go
What is OpenEdge RDBMS Table Partitioning?
OpenEdge Table Partitioning (v11.4 and later)

The OpenEdge RDBMS Table Partitioning feature allows you to organize the rows of a table into multiple physical storage objects (i.e. partitions), based on one or more column values, in an application-transparent manner.

By using this feature you can achieve increased data availability and make maintenance operations easier, quicker, and more efficient.

You can partition data of existing tables quickly and gradually move data into the new storage objects. When all are moved, truncate previous areas to recover disk space.
How to do it
partition setup: 4 possible ways

0) OpenEdge Explorer
1) OpenEdge Management
2) write programs to call 4GL API
3) SQL DDL !!!
partition setup: 4 possible ways

0) OpenEdge Explorer
1) OpenEdge Management
2) write programs to call 4GL API
3) SQL DDL !!!
Example 1: isports database
(order date range partitions)
steps:

0) enable partitioning
1) add areas
2) define partitions
3) move data
enable table partitioning

proutil foo -C enabletablepartitioning

Adding Table Partitioning file _Partition-Policy
Adding Table Partitioning file _Partition-Policy-Detail
Enable Table Partitioning successful.
Table Partitioning has been successfully enabled

proutil foo -C enabletpidxbuild

TP Index Rebuild has been enabled for \database foo. (12479)
define partitions

proserve foo -S 1234 -B 10000

sqlexp -db foo -S 1234 < part.sql
define partition to cover the existing data

```sql
set schema 'pub';
alter table pub.order
    partition by range "Order-Date"
    using table area "Customer/Order Area"
    using index area "Order Index Area"
    (partition "orderp1" values <= ( '12/31/2018' )
    )
    using index "Order-Date";
commit;
```
define partitions for old and new data

```
alter table pub.order
    prepare for split pro_initial
    (
        partition "orderp1" values <= ( '12/31/2016' )
            using table area "old_orders",
        partition "orderp2" values <= ( '12/31/2017' )
            using table area "2017_orders",
        partition "orderp3" values <= ( '12/31/2018' )
            using table area "2018_orders"
    );
commit;
quit;
```
Quiz:

Where are the data now?
```
prooutil foo -C partitionmanage view table order status

PROGRESS Partition View

Database: /home/gus/pug/orders/foo
Date: Mon Jun 27 14:40:07 2016

PARTITION STATUS
------------------
<table>
<thead>
<tr>
<th>Table</th>
<th>Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUB.Order</td>
<td></td>
</tr>
<tr>
<td>initial:0</td>
<td>207</td>
</tr>
<tr>
<td>orderp1:1</td>
<td>0</td>
</tr>
<tr>
<td>orderp2:2</td>
<td>0</td>
</tr>
<tr>
<td>orderp3:3</td>
<td>0</td>
</tr>
</tbody>
</table>
```
move the data into the partitions we defined

```
proutil foo -C partitionmanage split table \
  pub.order composite initial \ 
  useindex "Order-Date"
```
move the data into the partitions we defined

```bash
proutil foo -C partitionmanager split table \ 
    pub.order composite initial \ 
    useindex "Order-Date"
```

...  

Target partition: orderp1[1], records moved: 207.
Target partition: orderp2[2], records moved: 0.
Target partition: orderp3[3], records moved: 0.
Source partition: initial[0], contains no records.
Total records processed: 207.

END: Split Operation For Table pub.order[0]
Split Operation finished successfully. (17359)
Example 2: 650 GB pm database (customer number range partitions)
Partitioning procedure, part 1

- Generate dbanalys report
- Backup ?
- Enable table partitioning and partition index build
- Add areas and extents for partitions
- Designate tables as partitioned
- Define partitions
- Split data into partitions
- Rebuild or compress indexes
Partitioning procedure, part 2

- Generate partition manage view table status reports
- Drop now empty initial partitions
- Truncate empty areas
- Remove extents of empty areas
- Generate dbanalys report
- Compare before and after reports
- Mark some partitions read-only?
- Backup?
partition setup
enable table partitioning

```
proutil pm -C enabletablepartitioning

Adding Table Partitioning file _Partition-Policy
Adding Table Partitioning file _Partition-Policy-Detail
Enable Table Partitioning successful.
Table Partitioning has been successfully enabled

proutil pm -C enabletpidxbuild

TP Index Rebuild has been enabled for \ database pm. (12479)
```
4 simple steps:

0) add areas for new partitions
1) define composite partition for existing data
2) define new partitions
3) move data from existing to new partitions
DB_PATH=/opt/db/gus3/pm/data
echo "#" > add.st
for AREA_NUM in {201..210}
do
    PART_NR=$(( AREA_NUM - 200 ))
    AREA_NAME=areastats_tb_p${PART_NR}
    EXT_NAME=${DB_PATH}/pm_${AREA_NUM}.d1
    echo d "${AREA_NAME}\":${AREA_NUM},64\;8 ${EXT_NAME}\n    >> add.st
done
prostrct add pm add.st
partition setup
define partition for the existing data

```
set schema 'pub';
alter table pub.stats
    partition by range "s-mdba-site-id"
    using table area "Data-stats"
    using index area "Index-stats"
    (partition stats_p0 values <= (9999))
    using index "date-sample",
            "stats-date",
            "db-date-sample",
            "s-sample#";
commit;
quit;
```
FAIL !
define partition for the existing data

```
set schema 'pub';
alter table pub.stats
    partition by range "s-mdba-site-id"
    using table area "Data-stats"
    using index area "Index-stats"
    (partition stats_p0 values <= ( 9999 )
    using index "date-sample",
        "stats-date",
        "db-date-sample",
        "s-sample#";

    commit;
    quit;
```
define partition for the existing data

set schema 'pub';
alter table pub.stats
 partition by range "s-mdba-site-id"
 using table area "Data-stats"
 using index area "Index-stats"
(
  partition stats_p0 values <= ( 'zzzz' )
)
 using index "date-sample",
  "stats-date",
  "db-date-sample",
  "s-sample#" ;
commit;
quit;
set schema 'pub';
alter table pub.stats prepare for split pro_initial
    ( partition stats_p1 values <= ( '107' )
        using table area "stats_tb_p1"
        using index area "stats_ix_p1"
    );

define new target partitions
define new target partitions

set schema 'pub';
alter table pub.stats prepare for split pro_initial
(
    partition stats_p1 values <= ( '107' )
    using table area "stats_tb_p1"
    using index area "stats_ix_p1"
);

. . . . repeat for the other partitions . . . .

alter table pub.stats prepare for split pro_initial
(
    partition stats_p9 values <= ( 'zzzz' )
    using table area "stats_tb_p9"
    using index area "stats_ix_p9"
);
commit;
quit;
define new target partitions

set schema 'pub';
alter table pub.stats prepare for split pro_initial
(
    partition stats_p1 values <= ('107')
    using table area "stats_tb_p1"
    using index area "stats_ix_p1"
);

. . . . repeat for the other partitions . . . .

alter table pub.stats prepare for split pro_initial
(
    partition stats_p9 values <= ('zzzz')
    using table area "stats_tb_p9"
    using index area "stats_ix_p9"
);
commit;
quit;
partitions defined.
move existing data.
move data with split utility

```
proutil pm -C partitionmanage \
    split table areastats composite initial \
    useindex date-sample

proutil pm -C partitionmanage \
    split table stats composite initial \
    useindex date-sample
```
split utility output

BEGIN: Split Operation For Table areastats (17384)
    Source Partition initial[0]
    Target Partition AREASTATS_P1[1]
    . . .
    Target Partition AREASTATS_P9[9]
Index date-sample has been identified as the scanning index (useIndex).
A non-unique index has been selected as the useindex index.
Additional locking is required with the use of this index.
Number of Records per Transaction (recs): 100

Do you want to continue (y/n)?
1000000 records processed. (15165)
2000000 records processed. (15165)
    . . .
Total records processed: 1276802814.
END: Split Operation For Table areastats[0]
Split Operation finished successfully. (17359)
PROGRESS Partition View
Database: /opt/db/gus3/pm
Date: Thu Jun 4 12:55:17 2015
PARTITION STATUS
----------------------
<table>
<thead>
<tr>
<th>Table</th>
<th>Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUB.AreaStats</td>
<td>0</td>
</tr>
<tr>
<td>areastats_p1:1</td>
<td>54652873</td>
</tr>
<tr>
<td>areastats_p2:2</td>
<td>28465470</td>
</tr>
<tr>
<td>areastats_p3:3</td>
<td>56881593</td>
</tr>
<tr>
<td>areastats_p4:4</td>
<td>207241438</td>
</tr>
<tr>
<td>areastats_p5:5</td>
<td>159970866</td>
</tr>
<tr>
<td>areastats_p6:6</td>
<td>217269832</td>
</tr>
<tr>
<td>areastats_p7:7</td>
<td>390946904</td>
</tr>
<tr>
<td>areastats_p8:8</td>
<td>104965394</td>
</tr>
<tr>
<td>areastats_p9:9</td>
<td>56408444</td>
</tr>
</tbody>
</table>

proutil –C partitionmanage view table areastats status
delete original disk storage

prostrct remove pm d "data-areastats"
You must use the proutil truncate bi command before doing

proutil pm -C truncate bi
OpenEdge Release 11.5 as of Fri Dec  5 18:20:55 EST 2014

prostrct remove pm d "data-areastats"
/opt/db/gus2/db/pm_110.d2 successfully removed. (6968)
prostrct remove pm d "data-areastats"
/opt/db/gus2/db/pm_110.d1 successfully removed. (6968)
gus@bunker15:gus2 $
areastats table partitions

<table>
<thead>
<tr>
<th>partition</th>
<th>range</th>
<th>nr. of rows</th>
<th>extent size</th>
</tr>
</thead>
<tbody>
<tr>
<td>areastats_tb_p1</td>
<td>107</td>
<td>54,652,873</td>
<td>6.5 G</td>
</tr>
<tr>
<td>areastats_tb_p2</td>
<td>107–118</td>
<td>28,465,470</td>
<td>3.4 G</td>
</tr>
<tr>
<td>areastats_tb_p3</td>
<td>118–18</td>
<td>56,881,593</td>
<td>6.8 G</td>
</tr>
<tr>
<td>areastats_tb_p4</td>
<td>18–33</td>
<td>207,241,438</td>
<td>24.7 G</td>
</tr>
<tr>
<td>areastats_tb_p5</td>
<td>33–50</td>
<td>159,970,866</td>
<td>19.0 G</td>
</tr>
<tr>
<td>areastats_tb_p6</td>
<td>50–66</td>
<td>217,269,832</td>
<td>25.9 G</td>
</tr>
<tr>
<td>areastats_tb_p7</td>
<td>66–81</td>
<td>390,946,904</td>
<td>46.6 G</td>
</tr>
<tr>
<td>areastats_tb_p8</td>
<td>81–90</td>
<td>104,965,394</td>
<td>12.5 G</td>
</tr>
<tr>
<td>areastats_tb_p9</td>
<td>90–zzzz</td>
<td>56,408,444</td>
<td>6.72 G</td>
</tr>
</tbody>
</table>
Example 3: atm database
(area number list partitions)
(as used in magic tablemove test)
enable table partitioning

proutil atm -C enabletablepartitioning

Adding Table Partitioning file _Partition-Policy
Adding Table Partitioning file _Partition-Policy-Detail
Enable Table Partitioning successful. Table Partitioning has been successfully enabled

proutil atm -C enabletpidxbuild

TP Index Rebuild has been enabled for \ database atm. (12479)
partition setup
define partition for existing

set schema 'pub';
alter table pub.account
    partition by list area51
    using table area "accounts"
    (partition "part51" values in (0)
    using table area "accounts"
    )
    using index "area";
commit;
add move target partition

```
alter table pub.account
  add partition "part52" values in (1)
  using table area "area52";
commit;
quit;
```
Quiz:

Where are the data now?
move data
use a 4GL program, maybe like this

def var n as int no-undo.
def var cn as int no-undo.

find first customer.
cn = cust-num.
find last customer.
mainloop:
   do while (true) transaction:
      do n = 1 to 1000:
         customer.myarea = 1.
         if cust-num = cn then leave mainloop.
         find prev customer.
      end.
   end.
end.
Example 4: 650 GB pm database (customer number range partitions) (dump and load method)
5 simple steps:

0) dump,
1) make new db,
2) partition,
3) load,
4) build indexes
Full dump and load partitioning procedure

- Generate "before" dbanalys report
- Backup ?
- Dump data definitions from source
- Binary dump tables from source database
- Create new target database from empty
- Enable table partitioning and partitioned index build
- Load data definitions
- Define partitions on empty database
- Binary load the data
- Build the indexes - 1 partition at a time, or 1 table at a time
- Generate partitionmanage view table status reports
- Generate "after" dbanalys report
- Compare before and after reports
partition setup
define partition key

set schema 'pub';
alter table pub.stats
set partition using index
   "date-sample",
   "stats-date",
   "db-date-sample",
   "s-sample#" ;

commit;
quit;
define partition ranges

set schema 'pub';
alter table pub.stats
    partition by range "s-mdba-site-id"
    using table area "Data-stats"
    using index area "Index-stats"

(partition "stats_p1" values <= ( '107' )
    using table area "stats_tb_p1"
    using index area "stats_ix_p1","
define partition ranges

set schema 'pub';
alter table pub.stats
  partition by range "s-mdba-site-id"
  using table area "Data-stats"
  using index area "Index-stats"
(
  partition "stats_p1" values <= ( '107' )
  using table area "stats_tb_p1"
  using index area "stats_ix_p1",

  . . . . for the other partitions . . . .

  partition "stats_p9" values <= ( 'zzzz' )
  using table area "stats_tb_p9"
  using index area "stats_ix_p9"
)
;
commit;
quit;
load data

proutil pm -C load /opt/tmp/dump/AreaStats.bd \ 
  -i -B 81920 >>binload.log

echo "areastats loaded."

(repeat as needed for other tables)
don't forget to load the tables you *didn't* partition!
build indexes: (4 indexes, 9 partitions)

echo `date +"%H:%M:%S"` "bulding indexes for stats table"

for IX_NAME in "stats-date" "db-date-sample" "date-sample" "s-sample#"
do
  for P_NUM in {1..9}
do
    echo "building index ${IX_NAME}, partition ${P_NUM}""n    echo y | \n    proutil pm -C tpidxbuild table stats \n      index ${IX_NAME} partition STATS_P${P_NUM} \n        -i -TB 64 -TM 32 -TMB 32 -B 1000
  done
done

echo `date +"%H:%M:%S"` "stats table done"
4gl code to show partition objects for a table

```sql
find _file where _file-name = "stats".

for each _storageobject
    where _object-number = _file-num and _object-type = 1:

    find _partition_policy
    where _storageobject._object-number = _partition_policy._object-number.

    display _partition-policy-name _object-number _partitionid _Object-attrrib _object-state .

end.
```
4gl code to show partitions

for each _partition-policy NO-LOCK:
    display _Partition-Policy-Name label "PName" format "x(16)"
    _Num-Columns label "Cols" format ">9"
    _Has-Range label "Range"
    _partition-policy._object-number label "File#" format "->>,>>,9"
    _Column-Name[1] label "Cname" format "x(12)"
    _Column-Name[2] label "Cname" format "x(12)"
    _Column-Name[3] label "Cname" format "x(12)"
with title "Partition Policy".

for each _partition-policy-detail of _partition-policy NO-LOCK:
    display _partition-id label "PartId" format "->>,>>,9"
    _Partition-name label "Partition Name" format "x(12)"
    _Partition-Column-Value[1] label "V[1]"
    _Attributes[1] label "Attr[1]"
    _Attributes[2] label "Attr[2]"
with title "Partition Policy Detail".
end.
end.
partition setup times *

<table>
<thead>
<tr>
<th>operation</th>
<th>areastats</th>
<th>stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>table size</td>
<td>93.7 G</td>
<td>28.4 G</td>
</tr>
<tr>
<td>nr of rows</td>
<td>1,276,802,814</td>
<td>76,601,749</td>
</tr>
<tr>
<td>define partitions &amp; areas</td>
<td>1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>split into 9 parts</td>
<td>77 hours</td>
<td>9.2 hours</td>
</tr>
<tr>
<td>table.bd file size</td>
<td>110.4 G</td>
<td>29.5 G</td>
</tr>
<tr>
<td>binary dump</td>
<td>~ 1.25 hours</td>
<td>~ 0.4 hours</td>
</tr>
<tr>
<td>binary load</td>
<td>2.66 hours</td>
<td>0.31 hours</td>
</tr>
<tr>
<td>index rebuild table</td>
<td>3.2 hours</td>
<td>0.22 hours</td>
</tr>
<tr>
<td>index rebuild 9 partitions</td>
<td>4.3 hours</td>
<td>0.30 hours</td>
</tr>
<tr>
<td>pm view table status</td>
<td>956 seconds</td>
<td>35 seconds</td>
</tr>
</tbody>
</table>

* YMMV, mistakes, transportation, meals, and accommodations not included
### Partition Setup Times *

<table>
<thead>
<tr>
<th>Operation</th>
<th>Areastats</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table size</td>
<td>93.7 G</td>
<td>28.4 G</td>
</tr>
<tr>
<td>Nr of rows</td>
<td>1,276,802,814</td>
<td>76,601,749</td>
</tr>
<tr>
<td>Define partitions &amp; areas</td>
<td>1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Split into 9 parts</td>
<td>77 hours</td>
<td>9.2 hours</td>
</tr>
<tr>
<td>Table.bd file size</td>
<td>110.4 G</td>
<td>29.5 G</td>
</tr>
<tr>
<td>Binary dump</td>
<td>~ 1.25 hours</td>
<td>~ 0.4 hours</td>
</tr>
<tr>
<td>Binary load</td>
<td>2.66 hours</td>
<td>0.31 hours</td>
</tr>
<tr>
<td>Index rebuild table</td>
<td>3.2 hours</td>
<td>0.22 hours</td>
</tr>
<tr>
<td>Index rebuild 9 partitions</td>
<td>4.3 hours</td>
<td>0.30 hours</td>
</tr>
<tr>
<td>Pm view table status</td>
<td>956 seconds</td>
<td>35 seconds</td>
</tr>
</tbody>
</table>

*YMMV, mistakes, transportation, meals, and accommodations not included*
Partitioning is easy.
Try it when you get home.
You'll like it!
Answers

Email: gus@progress.com