Automatic Informix Range Interval Partitioning and Rolling Windows to Organize your data

by

Lester Knutsen

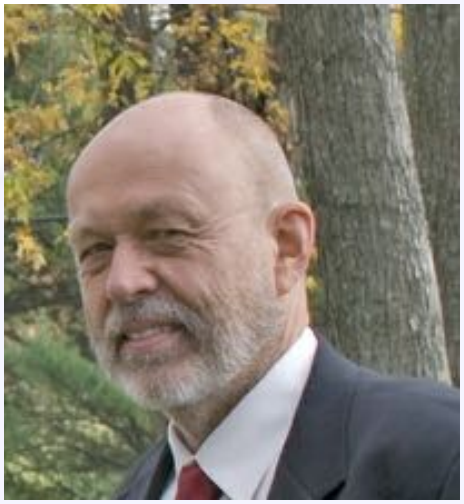
Webcast on June 21, 2018

At 2:00pm EDT

Advanced DataTools

or
Best Practices for Informix Partitioning

Lester Knutsen

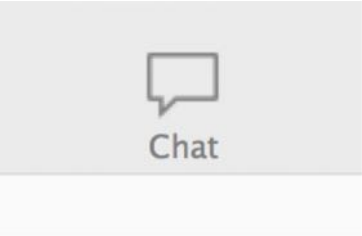


Lester Knutsen is President of Advanced DataTools Corporation, and has been building large data warehouse and business systems using Informix Database software since 1983. Lester focuses on large database performance tuning, training, and consulting. Lester is a member of the IBM Gold Consultant program and was presented with one of the Inaugural IBM Information Champion awards by IBM. Lester was one of the founders of the International Informix Users Group and the Washington Area Informix User Group.

lester@advancedatatools.com
www.advancedatatools.com
703-256-0267 x102

Webcast Guidelines

- The Webcast is being recorded. The Webcast replay and slides may be available in a few days.
- Please Mute your Line. Background sounds will distract everyone.
- Use the Chat Button in the upper right to ask questions.



Agenda

- What is Informix Partitioning?
- PDQ and Partitioning
- Automatic Informix Interval Partitioning
- Partitioning and SQL Explain Plans
- Informix Partition Rolling Windows
- Automatic Detach or Drop a Partition
- Manual Detach or Drop a Partition

Informix
Fragment = Partition

Note:

Fragmentation also mean too many extents and slow table access

Benefits of Partitioning

- Enable Parallel Database Queries
  - I/O can read Partitions at the same time
- Enable Fragment elimination in Queries
  - Less data to Read
- Enable Large Tables
  - Removes limits on number of pages
- Enables ease of Table Administration
  - Automatic Purging of old Data
- **Speed, Speed, Speed**

Partition Large Tables

- Any table greater then 2GB will benefit
- Any table larger then a typical chunk may benefit by being partitioned into multiple partitions or tablespaces
- Tables with more than 16,777,216 pages must be partitioned into multiple partitions or fragments
- Partition Tables for Performance

Partition Large Tables

- Avoid/alleviate the 16 million page partition limit
  - The maximum number of data pages in a partition is 2^24 -1 = 16,777,215 pages because the last 8 bits of the ROWID is the slot number of the row on the page

Partition Large Tables

- Allow parallel query
- Allow partition elimination
- More accurate data distributions for larger tables
- Storage placement of different classes of table content
  - Current rows on faster storage
  - Historical rows on slower/cheaper storage

Partition Large Tables

- Not required to have each Partition in a separate DBspace
- Each Partition creates a separate TableSpace for data
- Example:
  - Divide a Table into 12 Partitions in 4 DBspaces

Types of Informix Partitioning

- Round Robin
- Expression
- List
- Interval Range

Partition by Round Robin

- Simple
- Fastest insert speed for insert-heavy tables with many insert client sessions
- No partition elimination
- Adding or dropping partitions may require rewriting the entire table
- For tables only (not for index use)

Partition by Expression

- One or more BOOLEAN expressions defining what rows are to be placed in each partition
- Zero or One “REMAINDER” partition holding rows that do not match any partition expression
- Expressions are evaluated sequentially in the order defined.
- Long lists of expressions can be expensive to process at run time
- Only need to read the partition that holds the data for the expression

Partition by List

- BY LIST (column-expression)
  - PARTITION <name> VALUES (list), ...
  - PARTITION <name> IS NULL - optional
  - PARTITION <name> REMAINDER - optional
- Internally hashed so it can be faster to process than BY EXPRESSION partitioning when the expressions all refer to the same column(s)

Best Practices for Partition by Expression

- The Optimizer reads the table DDL from top to bottom, locate your most used Expressions at the Top
- Avoid Complex Expressions
  - Or clauses
  - Nested Expressions
- Keep it Simple

Automatic Partition by Interval Range

- Can use a numeric, DATE, or DATETIME column
- INTERVAL(<N units expression>)
  - STORE IN (<dbspace1>, <dbspace2>, ...) - optional
  - STORE IN (<function returning a dbspace name>) - optional
- PARTITION <name> VALUES < <number> IN <dbspace>

Partition by Range Example

```
create table testdata2 (
                    recno           serial,
                    recdate         date,
                    rectext         char(30),
                    recamount       dec(16,2)
)
partition by range (recdate)
interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
store in ( f1dbs, f2dbs, f3dbs, f4dbs )
partition p1 values < date ( "01/01/2012" )  in testdbs;
```

Partition by Range with Rolling Windows

- Range partition schemes can also automatically add and drop partitions to maintain a predefined number or range of active partitions

Rolling Partition by Range Example

```
create table testdata2 (
                    recno           serial,
                    recdate         date,
                    rectext         char(30),
                    recamount       dec(16,2)
)
partition by range (recdate)
interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
rolling ( 12 fragments ) detach
store in ( f1dbs, f2dbs, f3dbs, f4dbs )
partition p1 values < date ( "01/01/2012" )  in testdbs;
```

Partitioning and Parallel Database Query (PDQ)

- What is PDQ?
- Requirements for PDQ
- How do you configure PDQ?
- How do you manage PDQ?

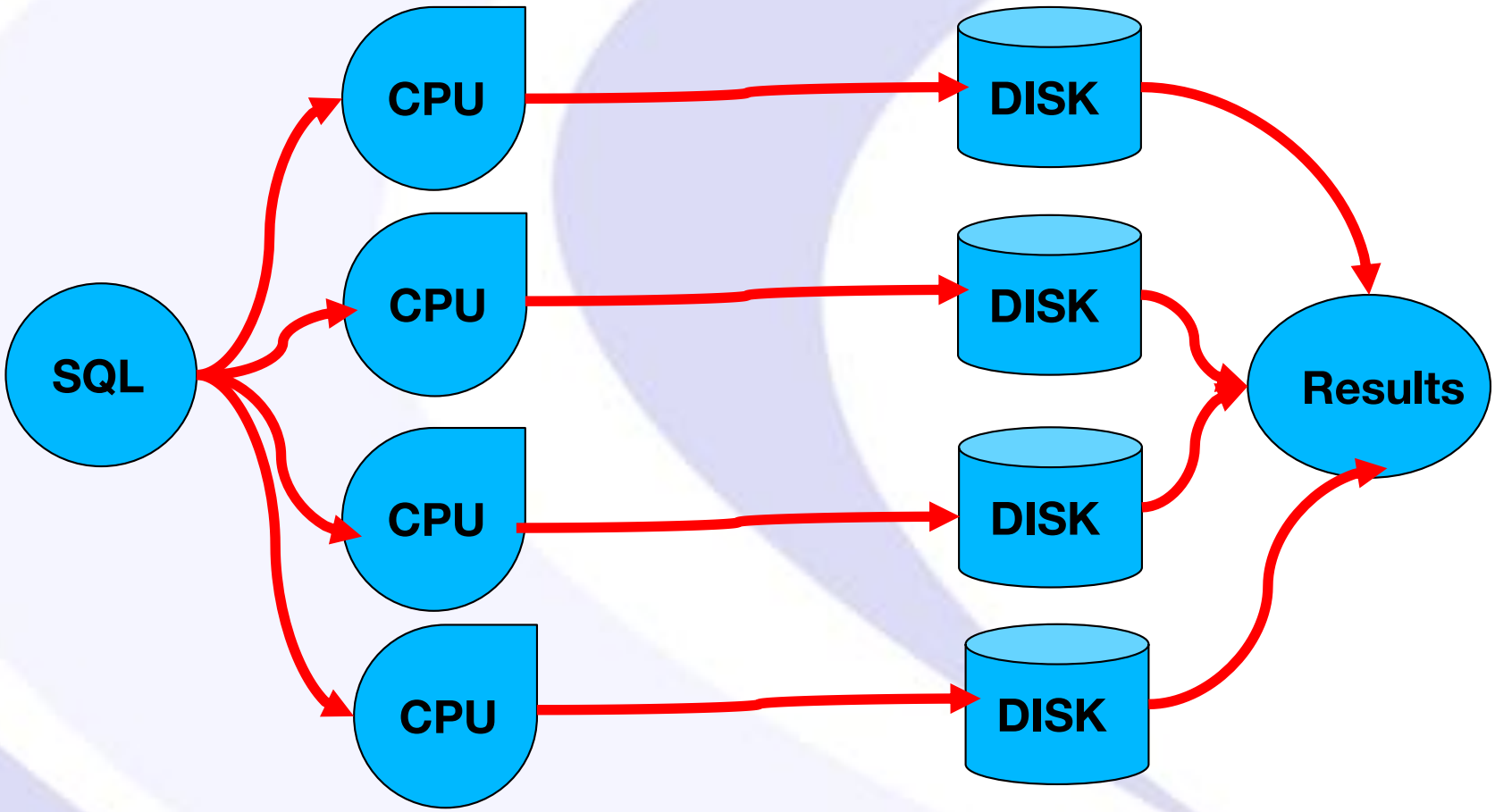
What is PDQ?

- PDQ divides large database queries into multiple parallel tasks.
- Parallelized PDQ operations include:
- Scans
- Sorts
- Joins
- Aggregates
- Inserts
- Deletes

Requirements for PDQ

- More then one CPU VP
- More then one DBSPACETEMP
- Table Partitioned or Fragmented
- Informix must be able to divide the SQL statement into multiple threads

PDQ



How Do You Enable PDQ?

- PDQ can be turned on and configured for a particular query or set of queries using the SQL statement:
  - SET PDQPRIORITY HIGH
  - SET PDQPRIORITY LOW
  - SET PDQPRIORITY <0 - 100>
- PDQ may be turned on for all queries run by particular users using the environmental parameter:
  - PDQPRIORITY
- Basic PDQ can be enabled with
  - SET PDQPRIORITY 1

How Do You Manage PDQ?

- PDQ administration involves managing resources allocated for parallel operations through Informix Server's:
  - ONCONFIG file configuration parameters
  - Memory Grant Manager
  - onmode

PDQ ONCONFIG Configuration Parameters

- MAX_PDQPRIORITY
  - System-wide governor of memory and CPU VP resources. Applied to PDQPRIORITY as a percentage multiplier
- DS_MAX_QUERIES
  - Maximum number of decision support queries that may run concurrently
- DS_TOTAL_MEMORY
  - Maximum memory available for all decision support queries.
- DS_MAX_SCAN
  - Maximum number of concurrent scan threads executed for parallel queries

Changing PDQ Parameters When the System is Online

- Change DS_TOTAL_MEMORY:
  - onmode -M kbytes
- Change DS_MAX_QUERIES:
  - onmode -Q max_queries
- Change MAX_PDQPRIORITY:
  - onmode -D priority
- Change DS_MAX_SCANS:
  - onmode -S max_number_scan_threads

Memory Grant Manager (MGM)

- The Memory Grant Manager manages and reserves resources for Parallel Database Queries including:
  - The number of concurrent queries
  - The number of scan threads
  - The number of PDQ threads
  - The amount of memory and CPU
- Monitor with onstat –g mgm

Factors to Consider in Tuning PDQ

- How many users will be running decision support queries at one time?
- Do some queries have higher priority than others?
- Should all DSS queries run at once, or should some wait?
- Is OLTP SQL running at the same time as DSS queries?
- How much memory is available?

Keys to PDQ

Must be able to process the data in parallel

- Tables must be partitioned (fragmented) on separate dbspaces and disk drives
- Must have multiple CPUVPs
- Have multiple temp dbspaces for sorting and grouping Data

Enabling Parallel Sorts

- PSORT_NPROCS – Number of CPUs to run sort threads on
- PSORT_DBTEMP – Number of tmp spaces to use for disk sorting
  - RAM Disk is best

PDQ Environment Variables Setup Example

- On an 8 CPU System

```
PDQPRIORITY=50
export PDQPRIORITY

PSORT_NPROCS=8
export PSORT_NPROCS

PSORT_DBTEMP=/dbtmp/t1:/dbtmp/t2:/dbtmp/t3:/dbtmp/t4:/dbtmp/t5
:/dbtmp/t6:/dbtmp/t7:/dbtmp/t8
export PSORT_DBTEMP
```

PDQ Environment Variables Setup Example

```
##################################################################
## Module: @(#)pdq.env     2.0     Date: 01/01/2015
## Author: Lester Knutsen  Email: lester@advancedatatools.com
##         Advanced DataTools Corporation
##################################################################

PDQPRIORITY=100
export PDQPRIORITY

PSORT_NPROCS=4
export PSORT_NPROCS

PSORT_DBTEMP=/tmp/t1:/tmp/t2:/tmp/t3:/tmp/t4
export PSORT_DBTEMP
```

Partition Examples

- Round Robin Partitioning
- Show Partition Information and Sysfragments
- Expression Partitioning
- Automatic Interval Partitioning
- Interval and Index Partitioning
- SQL Explain Output from Partitioning
- Rolling Windows Partitioning
- Automatic Detach or Drop Partitions
- Manual Detach or Drop Partitions

Example Scripts

- setup4.sh
- 00-createdb.sql
- 01-orders_base.sql
- 01-show_partition.sql
- 02-orders_rr.sql
- 02-show_partition.sql
- 03-orders_exp.sql
- 03-show_partition.sql
- 04-orders_interval.sql
- 04-show_partition.sql
- 05-examplequery.sql
- 05-orders_int_idx.sql
- 05-show_partition.sql
- 06-orders_rolling.sql
- 06-purge_function.sql
- 06-show_partition.sql
- 07-orders_serial.sql
- 07-show_partition.sql
- 08-mk_detach_partion.sql
- show_partition.sql

Script to Show Partitions (1/1)

```
-- ##################################################################
-- ## Module: @(#)01-show_partition.sql      2.0      Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##         Advanced DataTools Corporation (c)
-- ##################################################################

-- Create base table - no partitioning
update statistics ;

-- Count number of partations;
select t.tabname, f.fragtype, count(*) Partition_count
from systables t, sysfragments f
where t.tabid = f.tabid
and t.tabid > 99
and t.tabname in ("base_orders")
group by 1, 2
order by 1, 2;

-- Show Table partations
select t.tabname,
        t.tabid,
        f.fragtype,
        f.exprtext[1,100],
        f.partition
from systables t, sysfragments f
where t.tabid = f.tabid
and t.tabid > 99
and f.fragtype = "T"
and t.tabname in ("base_orders")
;

-- Show Index partations
select t.tabname,
        t.tabid,
        f.fragtype,
        f.exprtext[1,100],
        f.partition
from systables t, sysfragments f
where t.tabid = f.tabid
"01-show_partition.sql" [readonly] 54L, 1308C                    8,19          Top
```

Script to Show Partitions (2/2)

```
-- Show Index partations
select t.tabname,
        t.tabid,
        f.fragtype,
        f.exprtext[1,100],
        f.partition
from systables t, sysfragments f
where t.tabid = f.tabid
and t.tabid > 99
and f.fragtype = "I"
and t.tabname in ("base_orders")
;

-- Show ALL partations
select t.tabname,
        t.tabid,
        f.*
from systables t, sysfragments f
where t.tabid = f.tabid
and t.tabid > 99
and t.tabname in ("base_orders")
;
                                                    69,1          Bot
```

Sysfragment Table Key Fields

- Fragtype – T for table or I for Index
- Tabid – Id of the table in Systables
- Indexname – Index Name
- Partn – Partition Number on Disk
- Strategy
  - R – Round Roubin
  - E – Expression
  - I – In Dbspace
  - N – Interval or Rolling Windows
  - L – List
  - T – Table bases
  - H – Table Hierachry
- Exprtext – Text field with fragmentation plan

Sysfragments Table

```
DISPLAY:  Next  Restart  Exit
Display next page of results.

------------ testpartitions@train1 ------------- Press CTRL-W for Help --------


fragtype    T
tabid       102
indexname
colno       0
partn       9437188
strategy    E
location    L
servername
evalpos     7
exprtext
((order_date >= DATE ('08/01/2017' ) ) AND (order_date <= DATE ('08/31/2017' ) ) )
exprbin     <BYTE value>
exprarr     <BYTE value>
flags       0
dbspace     datab4ddbs
levels      0
npused      1.00000000000
nrows       23.00000000000
clust       0.00
partition   p_exp8
version     6
nupdates    0.00
ndeletes    0.00
ninserts    0.00
```

Partition by Round Robin

```
-- ##################################################################
-- ## Module: @(#)02-orders_rr.sql      2.0      Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##         Advanced DataTools Corporation (c)
-- ##################################################################

-- Create Partition by Round Robin

drop table if exists orders_rr;

create table orders_rr
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by round robin in datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs ;

insert into orders_rr select * from base_orders;

create unique index orders_rr_pk on orders_rr ( order_num ) in datab4adbs;

alter table orders_rr add constraint primary key ( order_num );


select count(*) from orders_rr;
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_rr
group by 1, 2
order by 1, 2;

select count(*) num_data_part
                                                                  1,1          Top
```

Advanced DataTools

Partition by Expression

```
-- Create Partition by Expression

drop table if exists orders_exp;

create table orders_exp
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by expression
partition p_exp1 order_date between  '01/01/2017' and '01/31/2017'  in datab4adbs,
partition p_exp2 order_date between  '02/01/2017' and '02/28/2017'  in datab4bdbs,
partition p_exp3 order_date between  '03/01/2017' and '03/31/2017'  in datab4cdbs,
partition p_exp4 order_date between  '04/01/2017' and '04/30/2017'  in datab4ddbs,

partition p_exp5 order_date between  '05/01/2017' and '05/31/2017'  in datab4adbs,
partition p_exp6 order_date between  '06/01/2017' and '06/30/2017'  in datab4bdbs,
partition p_exp7 order_date between  '07/01/2017' and '07/31/2017'  in datab4cdbs,
partition p_exp8 order_date between  '08/01/2017' and '08/31/2017'  in datab4ddbs,

partition p_exp9 order_date between  '09/01/2017' and '09/30/2017'  in datab4adbs,
partition p_exp10 order_date between  '10/01/2017' and '10/31/2017'  in datab4bdbs,
partition p_exp11 order_date between  '11/01/2017' and '11/30/2017'  in datab4cdbs,
partition p_exp12 order_date between  '12/01/2017' and '12/31/2017'  in datab4ddbs,

partition p_exp13 order_date between  '01/01/2018' and '01/31/2018'  in datab4adbs,
partition p_exp14 order_date between  '02/01/2018' and '02/28/2018'  in datab4bdbs,
partition p_exp15 order_date between  '03/01/2018' and '03/31/2018'  in datab4cdbs,
partition p_exp16 order_date between  '04/01/2018' and '04/30/2018'  in datab4ddbs
;
                                                                                    6,0-1        17%
```

Automatic Partition by Range

```
-- Create Partition by Range Interval

drop table if exists orders_interval;

create table orders_interval
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by range (order_date) interval ( NUMTOYMINTERVAL ( 1,'MONTH'))
store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
partition p1 values < date ( '01/01/2012' )  in datadbs
;


insert into orders_interval select * from base_orders;

create unique index orders_interval_pk on orders_interval ( order_num ) in datab4adbs;

alter table orders_interval add constraint primary key ( order_num );


select count(*) from orders_interval;
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
group by 1, 2
order by 1, 2;

select count(*) num_data_part
from sysfragments where tabid = (select tabid from systables where tabname = 'orders_interval')
and fragtype = 'T';
                                                                                    42,0-1        60%
```

Advanced DataTools

Indexes and Partitioned Tables

- For Best Performance with Detach or Drop Partitions - Indexes should be Partitioned using the same method as the table
- Otherwise need to drop or disable indexes to detach or drop Partitions
- Add the Partition Field to the end of the Index to avoid this problem
- Done Automatically for Rolling Windows

Automatic Partition by Interval with Partitioned Index

```
-- ##################################################################
-- ## Module: @(#)05-orders_int_idx.sql      2.0      Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##          Advanced DataTools Corporation (c)
-- ##################################################################

-- Create Partition by Range Interval with Index Partitioned

drop table if exists orders_interval;

create table orders_interval
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by range (order_date) interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
partition p1 values < date ( "01/01/2012" )  in datadbs
;


insert into orders_interval select * from base_orders;

create unique index orders_interval_pk on orders_interval ( order_num, order_date )
partition by range (order_date) interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
partition p1 values < date ( "01/01/2012" )  in datadbs;

alter table orders_interval add constraint primary key ( order_num );

select count(*) from orders_interval;
                                                                    1,1          Top
```

Partitions and the SQL Explain Plan

- The SQL Explain Plan will show you which Partitions or ALL that are access to process an SQL Statement
- Serial, Fragments: ALL
  - SEQUENTIAL SCAN (Serial, fragments: ALL)
- Serial, Fragments: Fragment Number
  - SEQUENTIAL SCAN (Serial, fragments: 73)
- Parallel, Fragments: ALL
  - SEQUENTIAL SCAN (Parallel, fragments: ALL)
- Parallel, Fragments: Fragment Number
  - SEQUENTIAL SCAN (Parallel, fragments: 73)

SQL Explain Script

```
-- ##################################################################

set explain on;

-- set PDQPRIORITY 1;

-- Read all Partitions
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
group by 1, 2
order by 1, 2;

-- Read one Partition
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
where year(order_date) = 2010
and month(order_date) = 2
group by 1, 2
order by 1, 2;

-- Read one Partition
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
where order_date between "01/01/2010" and "01/31/2010"
group by 1, 2
order by 1, 2;

-- Read One Record
select *
from orders_interval
where order_num = 300
order by 1, 2;

-- Read One Record
select *
from orders_interval
where order_num = 300
and order_date = "02/01/2010"
order by 1, 2;
                                                                43,1          Bot
```

SQL Explain – No PDQ

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:37)
------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
group by 1, 2
order by 1, 2

Estimated Cost: 495
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By  Group By

 1) informix.orders_interval: SEQUENTIAL SCAN  (Serial, fragments: ALL)


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     322        322       322        00:00.00   26

  type     rows_prod  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  group    14         1         322        00:00.00   470

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  sort     14         1         14         00:00.00   0


QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:37)
------
"sqexplain.out.save" 375L, 11677C                                    13,1          Top
```

SQL Explain – No PDQ

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:37)
------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
where year(order_date) = 2018
and month(order_date) = 2
group by 1, 2
order by 1, 2

Estimated Cost: 30
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By  Group By

  1) informix.orders_interval: SEQUENTIAL SCAN  (Serial, fragments: ALL)

        Filters: (MONTH (informix.orders_interval.order_date ) = 2 AND YEAR (informix.orders_interval.order_date ) = 201
8 )


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ------------------------------------------------------------------
  scan     t1     23         3         322        00:00.00   26

  type     rows_prod  est_rows  rows_cons  time       est_cost
  ------------------------------------------------------------
  group    1          1         23         00:00.00   5

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ------------------------------------------------------------
  sort     1          1         1          00:00.00   0
                                                                                    75,1          10%
```

SQL Explain – No PDQ

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:37)
------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
where order_date between "01/01/2018" and "01/31/2018"
group by 1, 2
order by 1, 2

Estimated Cost: 48
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By  Group By

  1) informix.orders_interval: SEQUENTIAL SCAN  (Serial, fragments: 73)
        Fragments Scanned: (73) sys_p73 in datab4adbs

        Filters: (informix.orders_interval.order_date <= 01/31/2018 AND informix.orders_interval.order_date >= 01/01/201
8 )


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     23         36        23         00:00.00   3

  type     rows_prod  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  group    1          1         23         00:00.00   46

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
                                                        77,0-1        22%
```

Notice Date Format

Advanced DataTools

SQL Explain – No PDQ

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:37)
-------
select *
from orders_interval
where order_num = 300
order by 1, 2

Estimated Cost: 1
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By

  1) informix.orders_interval: INDEX PATH

    (1) Index Name: informix. 104_15
        Index Keys: order_num   (Serial, fragments: ALL)
        Lower Index Filter: informix.orders_interval.order_num = 300


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     1          1         1          00:00.00   1

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  sort     1          1         1          00:00.00   0


QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:39)
-------
                                                                155,1         34%
```

SQL Explain – No PDQ

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:52:39)
------
select *
from orders_interval
where order_num = 300
and order_date = "02/01/2018"
order by 1, 2

Estimated Cost: 1
Estimated # of Rows Returned: 1

  1) informix.orders_interval: INDEX PATH

    (1) Index Name: informix.orders_interval_pk
        Index Keys: order_num order_date   (Serial, fragments: 74)
        Fragments Scanned: (74) sys_p74 in databb4ddbs
        Lower Index Filter: (informix.orders_interval.order_num = 300 AND informix.orders_interval.order_date = 02/01/20
18 )


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time      est_cost
  ---------------------------------------------------------------------
  scan     t1     1          1         1          00:00.00  1


QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:00)
------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
                                                                    190,1         45%
```

Notice Date Field

SQL Explain – PDQ Set

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:00)
------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
group by 1, 2
order by 1, 2

Estimated Cost: 471
Estimated # of Rows Returned: 1
Maximum Threads: 16
Temporary Files Required For: Order By  Group By

  1) informix.orders_interval: SEQUENTIAL SCAN  (Parallel, fragments: ALL)


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     322        322       322        00:00.00   2

  type     rows_prod  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  group    14         1         322        00:00.00   470

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  sort     14         1         14         00:00.00   0

QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:00)
                                                                224,1         55%
```

SQL Explain – PDQ Set

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:00)
-------
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_interval
where order_date between "01/01/2018" and "01/31/2018"
group by 1, 2
order by 1, 2

Estimated Cost: 48
Estimated # of Rows Returned: 1
Maximum Threads: 1
Temporary Files Required For: Order By  Group By

  1) informix.orders_interval: SEQUENTIAL SCAN  (Parallel, fragments: 73)
        Fragments Scanned: (73) sys_p73 in datab4adbs

        Filters: (informix.orders_interval.order_date <= 01/31/2018 AND informix.orders_interval.order_date >= 01/01/201
8 )


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     23         36        23         00:00.00   3

  type     rows_prod  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  group    1          1         23         00:00.00   46
                                                                  300,0-1       77%
```

Notice Date Field

SQL Explain – PDQ Set

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:00)
-------
select *
from orders_interval
where order_num = 300
order by 1, 2

Estimated Cost: 1
Estimated # of Rows Returned: 1
Maximum Threads: 15
Temporary Files Required For: Order By

  1) informix.orders_interval: INDEX PATH

    (1) Index Name: informix.orders_interval_pk
        Index Keys: order_num order_date   (Parallel, fragments: ALL)
        Lower Index Filter: informix.orders_interval.order_num = 300


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  ---------------------------------------------------------------------
  scan     t1     1          1         1          00:00.00   1

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ---------------------------------------------------------------
  sort     1          1         1          00:00.00   0


QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:01)
                                                          343,1          90%
```

SQL Explain – PDQ Set

```
QUERY: (OPTIMIZATION TIMESTAMP: 06-20-2018 14:53:01)
------
select *
from orders_interval
where order_num = 300
and order_date = "02/01/2018"
order by 1, 2

Estimated Cost: 1
Estimated # of Rows Returned: 1
Maximum Threads: 1

  1) informix.orders_interval: INDEX PATH

    (1) Index Name: informix.orders_interval_pk
        Index Keys: order_num order_date   (Parallel, fragments: 74)
        Fragments Scanned: (74) sys_p74 in databa4ddbs
        Lower Index Filter: (informix.orders_interval.order_num = 300 AND informix.orders_interval.order_date = 02/01/20
18 )


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                orders_interval

  type     table  rows_prod  est_rows  rows_scan  time      est_cost
  ---------------------------------------------------------------------
  scan     t1     1          1         1          00:00.00  1


                                                                    378,0-1       Bot
```

Notice Date Field

Automatic Rolling Interval Partition

```
-- ##################################################################
-- ## Module: @(#)06-orders_rolling.sql      2.0      Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##          Advanced DataTools Corporation (c)
-- ##################################################################

-- Create Rolling Windows Partition by Range Interval with Index Partitioned

drop table if exists orders_rolling ;

create table orders_rolling
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by range (order_date) interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
rolling ( 4 fragments ) detach
store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
partition p1 values < date ( "01/01/2017" )  in datadbs
;


insert into orders_rolling select * from base_orders;

create unique index orders_rolling_pk on orders_rolling ( order_num, order_date ) ;
-- partition by range (order_date) interval ( NUMTOYMINTERVAL ( 1,"MONTH"))
-- rolling ( 4 fragments ) detach
-- store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
-- partition p1 values < date ( "01/01/2012" )  in datadbs;

-- alter table orders_rolling add constraint primary key ( order_num, order_date );
"06-orders_rolling.sql" [readonly] 53L, 1878C                                  2,35          Top
```

Rolling Window

DB Scheduler – Purge Partition Task

```
DISPLAY:  Next  Restart  Exit
Display next page of results.

------------------------ sysadmin@train1 -------- Press CTRL-W for Help --------


tk_id              38
tk_name            purge_tables
tk_description     Daily task to ensure that rolling window tables stay within
                    limits
tk_type            TASK
tk_sequence        13
tk_result_table
tk_create
tk_dbs             sysadmin
tk_execute         rwt_purge_tables
tk_delete
tk_start_time      00:45:00
tk_stop_time
tk_frequency         1 00:00:00
tk_next_execution  2018-06-21 00:45:00
tk_total_executio+ 13
tk_total_time      0.793486420803
tk_monday          t
tk_tuesday         t
tk_wednesday       t
tk_thursday        t
tk_friday          t
tk_saturday        t
tk_sunday          t
tk_attributes      412
tk_group           TABLES
tk_enable          t
tk_priority        0
```

Purge Partition Task Function Not Documented

```
-- ################################################################
-- ## Module: @(#)06-purge_function.sql      2.0      Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##         Advanced DataTools Corporation (c)
-- ################################################################

-- This is undocumented and may change or not work or create other problems
-- Use with caution and at your own Risk!!!

-- This is the function the Informix 12.10.FC10 scheduler calls when it runs
-- purge_tables task to drop or detach rolling window partitions that
-- are out of scope. The info is from scripts in $INFORMIXDIR/etc/sysadmin

execute function sysadmin:exectask ("db_purge_tables", "testpartitions" );
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"06-purge_function.sql" [readonly] 14L, 748C                    12,49          All
```

Manual Partition Purge Script

```
-- Creat a temp table to cast the sysfragment.exprtext from text to character

drop table if exists tmp_partition_name;

create temp table tmp_partition_name (
        tabname         varchar(128),
        tabid           integer,
        fragtype        char(1),
        fragexpression  char(1000),
        partition       varchar(128)
        );

-- Insert the rows from sysfragments
unload to /tmp/sysfragments.uld
select t.tabname,
        t.tabid,
        f.fragtype,
        f.exprtext[1,1000],
        f.partition
from systables t, sysfragments f
where t.tabid = f.tabid
and t.tabid > 99
and f.fragtype = "T"
and f.partition matches "sys*" -- System generated partition name
and t.tabname in ( "orders_interval" )
;

load from /tmp/sysfragments.uld
insert into tmp_partition_name;

unload to "detach_partition.sql" delimiter ";"
select "alter fragment on table " || trim(tabname) || " detach partition " ||
        trim(partition) || " "  || trim(tabname) || "_" || trim(partition) ,
        " in " || trim(fragexpression)
from tmp_partition_name
where fragexpression matches "*2017*";
-- where fragexpression matches "*02/01/2017*";

                                                                46,0-1        Bot
```

Create Temp Table to Parse Partitions

Create SQL To Detach Partitions

SQL Script Output

```
alter fragment on table orders_interval detach partition sys_p61 orders_interval_sys_p61;-- VALUES >= DATE ('01/01/2017'
 ) AND VALUES < DATE ('02/01/2017' );
alter fragment on table orders_interval detach partition sys_p62 orders_interval_sys_p62;-- VALUES >= DATE ('02/01/2017'
 ) AND VALUES < DATE ('03/01/2017' );
alter fragment on table orders_interval detach partition sys_p63 orders_interval_sys_p63;-- VALUES >= DATE ('03/01/2017'
 ) AND VALUES < DATE ('04/01/2017' );
alter fragment on table orders_interval detach partition sys_p64 orders_interval_sys_p64;-- VALUES >= DATE ('04/01/2017'
 ) AND VALUES < DATE ('05/01/2017' );
alter fragment on table orders_interval detach partition sys_p65 orders_interval_sys_p65;-- VALUES >= DATE ('05/01/2017'
 ) AND VALUES < DATE ('06/01/2017' );
alter fragment on table orders_interval detach partition sys_p66 orders_interval_sys_p66;-- VALUES >= DATE ('06/01/2017'
 ) AND VALUES < DATE ('07/01/2017' );
alter fragment on table orders_interval detach partition sys_p67 orders_interval_sys_p67;-- VALUES >= DATE ('07/01/2017'
 ) AND VALUES < DATE ('08/01/2017' );
alter fragment on table orders_interval detach partition sys_p68 orders_interval_sys_p68;-- VALUES >= DATE ('08/01/2017'
 ) AND VALUES < DATE ('09/01/2017' );
alter fragment on table orders_interval detach partition sys_p69 orders_interval_sys_p69;-- VALUES >= DATE ('09/01/2017'
 ) AND VALUES < DATE ('10/01/2017' );
alter fragment on table orders_interval detach partition sys_p70 orders_interval_sys_p70;-- VALUES >= DATE ('10/01/2017'
 ) AND VALUES < DATE ('11/01/2017' );
alter fragment on table orders_interval detach partition sys_p71 orders_interval_sys_p71;-- VALUES >= DATE ('11/01/2017'
 ) AND VALUES < DATE ('12/01/2017' );
alter fragment on table orders_interval detach partition sys_p72 orders_interval_sys_p72;-- VALUES >= DATE ('12/01/2017'
 ) AND VALUES < DATE ('01/01/2018' );
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"detach_partition.sql" [readonly] 12L, 1896C                    1,1          All
```

Automatic Partition by Interval on Serial Column

```
-- ##################################################################
-- ## Module: $(#)07-orders_serial.sql      2.0     Date: 06/01/2018
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##         Advanced DataTools Corporation (c)
-- ##################################################################

-- Create Serial Number Rollowing Windows Partition by Range Interval with Index Partitioned

drop table if exists orders_serial;

create table orders_serial
  (
    order_num serial not null ,
    order_date date,
    customer_num integer not null ,
    ship_instruct char(40),
    backlog char(1),
    po_num char(10),
    ship_date date,
    ship_weight decimal(8,2),
    ship_charge money(6,2),
    paid_date date
  )
partition by range (order_num) interval (1000)
rolling ( 100 fragments ) detach
store in ( datab4adbs, datab4bdbs, datab4cdbs, datab4ddbs )
partition p1 values < 0  in datadbs
;


insert into orders_serial select * from base_orders;

create unique index orders_serial_pk on orders_serial ( order_num ) ;

alter table orders_serial add constraint primary key ( order_num );

select count(*) from orders_serial;
select year(order_date) year, month(order_date) month,  count(*) rec_count
from orders_serial
"07-orders_serial.sql" [readonly] 49L, 1579C                    2,34          Top
```

Serial Interval

Questions?

Send follow-up questions to
lester@advancedatatools.com

Next Webcast
Informix Best Practices

- **Uninterruptible Informix Database transactions are here!**
  - **by Art Kagel - July 19, 2018 at 2:00pm EST**
  - Have you ever had a large batch job that had to be rerun from scratch when the primary server crashed after the job had been running for many hours? Wouldn't you like to have that job just continue running when your secondary server takes over? You can!
  - This will be a live demonstration of how to configure your servers to take advantage of the Informix Transaction Survival feature.
- **Unloading and Loading data with Informix Best Practices**
  - **by Jack Parker - August 23, 2018 at 2:00pm EST**
  - Whether you're reorganizing a table, loading a table or doing a full data migration, you will want to use the unload/load capabilities of Informix. This is an in depth look at the various options available to you in such an endeavor as well as issues which should be taken into consideration.

Please register for each webcast here at:

http://advancedatatools.com/Informix/NextWebcast.html

Informix Training 2018

- **Advanced Informix Performance Tuning**
  - February 5-8, 2018 – **Completed and Filled up**
- **Informix for Database Administrators**
  - May 21-24, 2018 - **Completed and Filled up**
  - September 17-20, 2018 – **Registration in Progress**

- All courses can be taken online on the web from your desk or at our training center in Virginia.
- We guarantee to *NEVER* cancel a course and will teach a course as long as one student is registered!
- Please register early as the last two courses have filled up and we have not been able to accommodate everyone.

http://advancedatatools.com/Training/InformixTraining.html

Eight New Training Servers

Each Student in class will have a server running Informix 12.10 with:

- 8 CPU Cores
- 16 GB RAM
- 1 SSD Disk
- 1- 4 disks

Informix Support and Training from the Informix Champions!

Advanced DataTools is an Advanced Level IBM Informix Data Management Partner, and has been an authorized Informix partner since 1993. We have a long-term relationship with IBM, we have priority access to high-level support staff, technical information, and Beta programs. Our team has been working with Informix since its inception, and includes 8 Senior Informix Database Consultants, 4 IBM Champions, 2 IIUG Director's Award winners, and an IBM Gold Consultant. We have Informix specialists Lester Knutsen and Art Kagel available to support your Informix performance tuning and monitoring requirements!

- ***Informix Remote DBA Support Monitoring***
- ***Informix Performance Tuning***
- ***Informix Training***
- ***Informix Consulting***
- ***Informix Development***

Free Informix Performance Tuning Webcast replays at:

http://advancedatatools.com/Informix/Webcasts.html

Email: info@advancedatatools.com

Web: http://www.advancedatatools.com

Thank You

Advanced DataTools Corporation

For more information:

Lester@advancedatatools.com

http://www.advancedatatools.com