Advanced DataTools Webcast

from the IBM Informix Champions

Database Indexes – Best Practices for Informix DBAs by Lester Knutsen

Thursday, August 29, 2019

at 2:00 pm EDT

Advanced DataTools

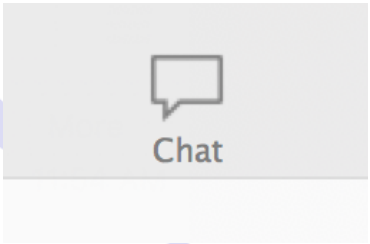
Lester Knutsen

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 will 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.



Database Indexes
Best Practices for Informix DBAs

- - What is an index?
- - How do you improve performance with an index?
- - What is the performance cost of an index?
- - How do you speed up creating indexes?
- - How do locks affect indexes?
- - How to monitor an index to see if it is needed or used?
- - When is an index unnecessary?
- - When is an index needed or required?
- - What is a clustered index?
- - How do you maintain indexes for peak efficiency?
- - What is a functional index?

Database Indexes

- Indexes are a data structure that improves the speed of data retrieval on a table
- Indexes…cost additional writes and storage space…
- Indexes are used to quickly locate data without having to search every row in a table
  - Source: https://en.wikipedia.org/wiki/Database_index

Demo: Performance with an Index

Example – five update statements like this:

```
update benchmark set price = price +
                ( select price from zip where benchmark.zip = zip.zip )
where id between 1 and 50000;
```

	No Index on Zip	Index on Zip
Real Time	9m38.010s	0m2.503s
Number Rows Scanned	8,778,469,012	709,542
SQL Estimated Costs	1,304,497	433,074

Types of Informix Indexes

- B-Tree Index – most common – used for all built-in data types
- Forest of Trees Index – B-Tree index with multiple root nodes
- R-Tree Index – Used for Spatial and two- or three- dimensional data
- Functional Index – provided by DataBlade module or indexes based on User Defined Functions

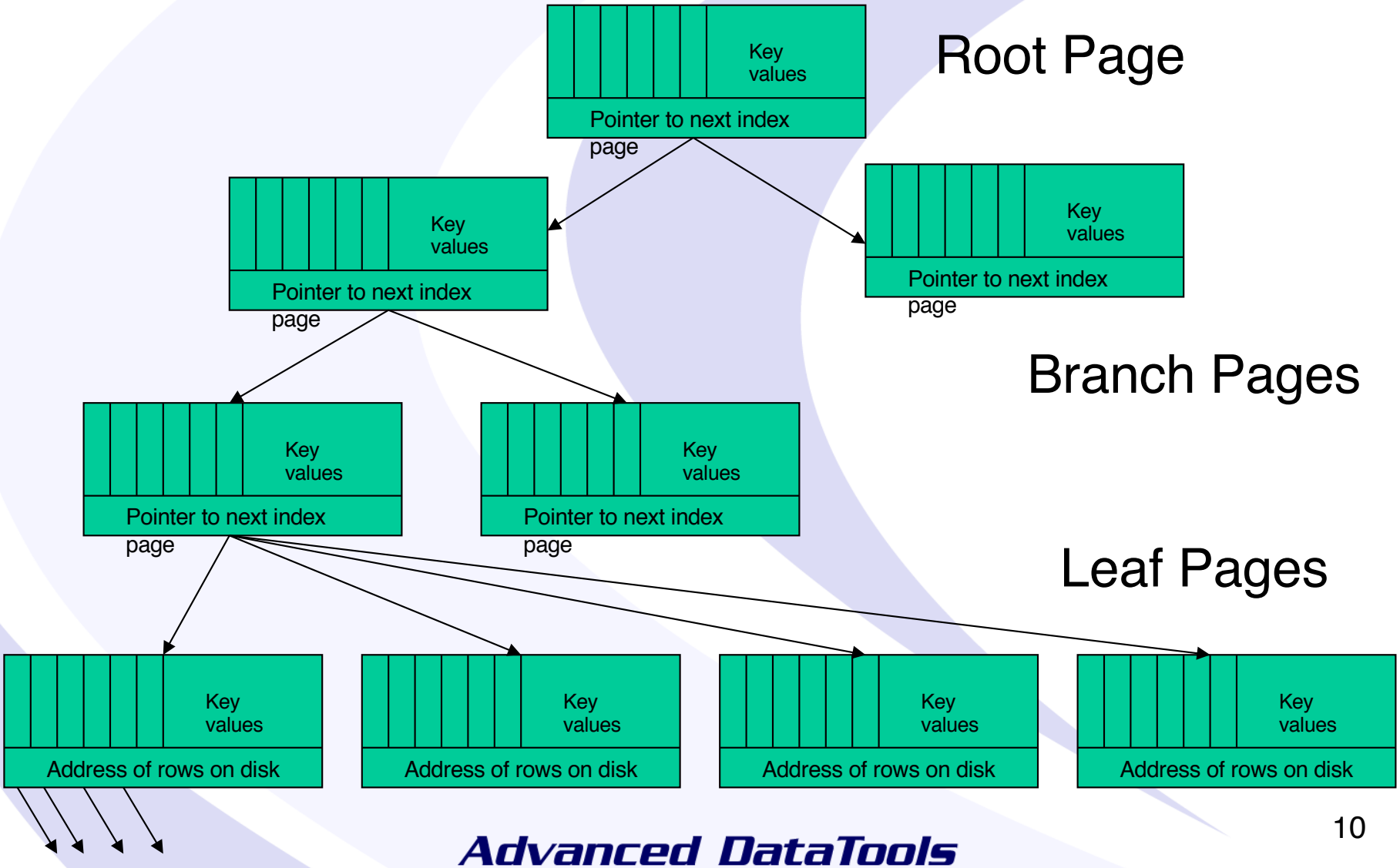
What is a Sequential Scan?

- Sequential Scan is how a table is read without an index
- Starts at the first row of the table and reads every row
- Reads in the order pages are stored on disk (random order)
- May be best performance when reading more than 20-25% of the rows in a table

What is a B-tree Index?

- A sorted hierarchy structure that contains pointers to the rows on disk to speed up access
- Like an index in a book, look up a term and it will point you directly to the page number

Structure of a B-Index



Structure of a B-Index

- Root page – points to branch pages
- Levels of branch pages – point to other branch pages until the bottom level is reached; this is a leaf page.
- Leaf pages point to where the row is located on disk (row id or partition and row id)
- The levels and size of an index depend on:
  - the number of unique keys in an index
  - the number of index entries that each page can hold
  - the size of the columns being indexed
- Index leaf pages contain:
  - Sorted column values – the key
  - Pointer to the data row – the address of the row
- A unique index contains one index entry for every row in the table.

Estimate Size of an Index

- Estimating conventional index pages:
  - https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.perf.doc/ids_prf_365.htm
- Formula for estimating index size = (((Column Size + 4 bytes) * percent unique) + row pointer + 4 bytes) * number of rows
  - percent unique is how unique the index is where 1 is unique index and .5 is 50% duplicates
  - Row pointer is 5 for non-partitioned tables and 9 for partitioned tables

Create and Drop Syntax

- Create index index_name on table (column_list);
  - create index if not exists idx_bills_customer_number on bills (customer_number);
- Drop index index_name;
  - drop index if exists idx_bills_customer_number;

Create and Drop Indexes Online – No Exclusive Locks

- Create index index_name on table (column_list) online;
  - create index if not exists idx_bills_customer_number on bills (customer_number) online;
- Drop index index_name online;
  - drop index if exists idx_bills_customer_number online;

Example – Performance of a Scan vs Index

Performance of an Index vs Scans

- Example – 100,000 row table with 4 index levels
- A sequential scan of the table will take 100,000 reads (1 read per row)
- An Index will take 4 reads for each level and 1 read for the row (5 reads per row)

Performance of an Index vs Scans

- How many Index I/O reads are required to:
  - Read 1 row – 5 reads
  - Read 10,000 rows – 50,000 reads
  - Read 20,000 rows – 100,000 reads
  - Read 50,000 rows – 250,000 reads
- At what point is a Scan less I/O reads than using an Index?

Performance Costs of an Index - Selects

- Select a row using an index – will require reading several index levels to get to the data row page

Performance Costs of an Index - Inserts

- Insert a row and new index entries must be added to the index
- If space does not exist on existing pages, new index pages and levels may need to be created
- Index Pages Splits can be expensive
- Look at Fill Factor when creating an Index
- Performance can be improved for large loads by disabling Indexes and then rebuilding the indexes after the load

Performance Costs of an Index - Deletes

- Delete a row and all index entries that point to the row must be deleted
- First the index entries are marked for deletion
- Next, in the background, the BTSCANNER goes through and deletes the entries based on ONCONFIG settings
  - **BTSCANN**ER num=1,threshold=5000,rangesize=-1,alice=6,compression=default
- This may require re-writing several index pages

Performance Costs of an Index - Updates

- Update a row and any columns that change and have index entries that point to the row will need to be updated
- An update is performed as a delete from the old index page, then an insert into the new page
  - Cost of a delete
  - Cost of an insert

Performance Costs of an Index - Summary

- When a row is inserted or deleted at random, allow three to four added page I/O operations per index.
- When a row is updated, allow six to eight page I/O operations for each index that applies to an altered column.
- When a transaction is rolled back, all this work must be undone. For this reason, rolling back a transaction can take a long time.

How Does an Index Affect Locks?

- Lock on a row will lock the related indexes
- A table with 5 indexes will generate 6 locks
- Insert, update, or delete on a row will automatically create a lock on the index key
- Tables with page-level locking will hold locks on the related index page

Indexes that are NOT Required Degrade Performance

- More I/O for inserts, deletes, and updates
- Unnecessary locks
- Drop all unnecessary indexes

When is an Index Required?

- Enforce unique columns in a table
- Primary keys
- Foreign keys

When is an Index Required?

- May improve performance of Joins
- May improve performance of Queries
- May improve performance of Sorts

When are Indexes Helpful?

- Columns used in joins
- Columns used in filter expressions (SQL where clause)
- Columns used for sorts, ordering, or grouping

When is an Index Unnecessary?

- On columns not used in:
  - Where statements
  - Order by statements
  - Join statements
  - Primary Keys or Constraints

Auto Index – System Generated Temp Indexes

- The Informix Server Optimizer may decide to automatically create a temporary index to execute a SQL statement
- When the SQL statement is complete the index is dropped
- Review SQL Explain Plans

```
2) informix.product: AUTOINDEX PATH
        Filters:
        Table Scan Filters: informix.product.product_number IN (1 , 2 )
    (1) Index Name: (Auto Index)
        Index Keys: product_code
```

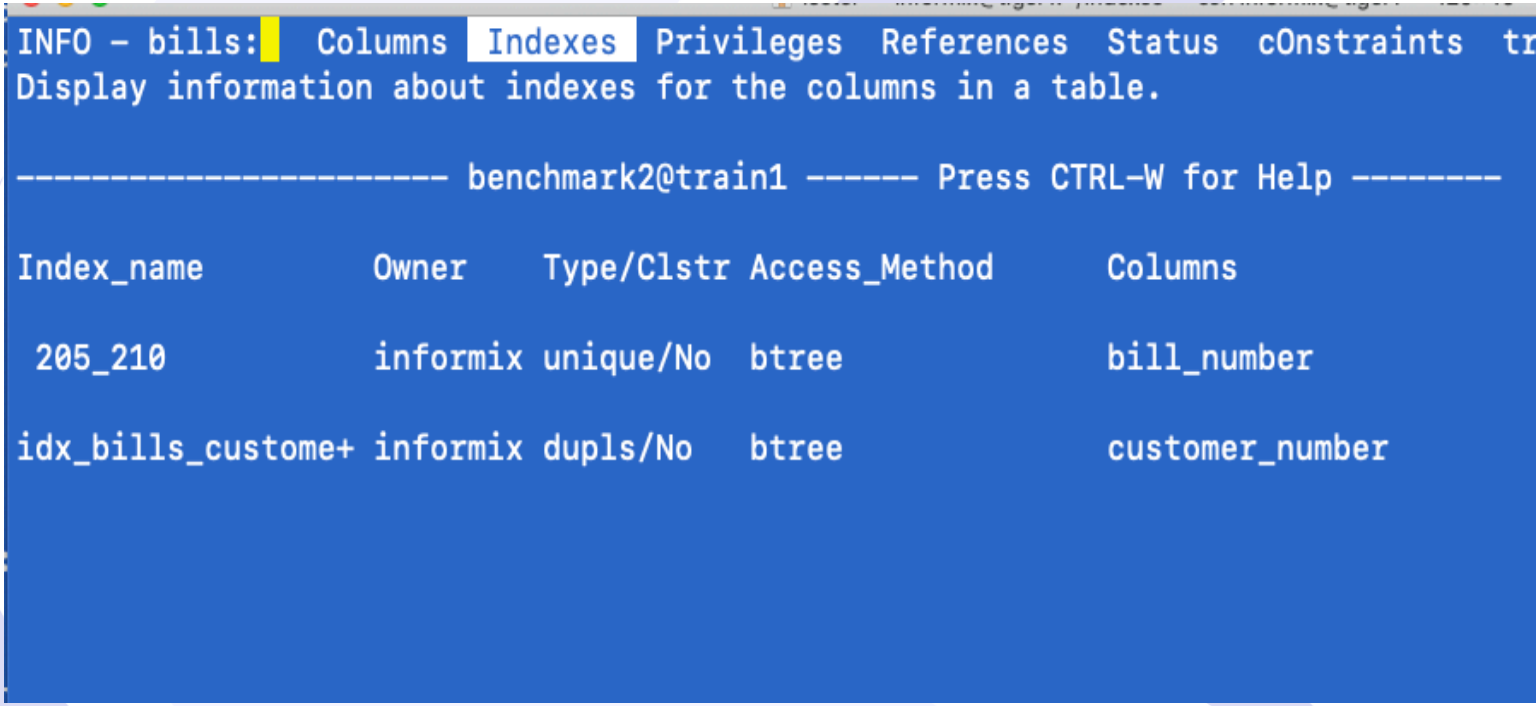
Auto Index in SQL Explain Plan

```
1) informix.customer: SEQUENTIAL SCAN

      Filters: informix.customer.start_date >= 01/01/2000

2) informix.product: AUTOINDEX PATH

      Filters:
      Table Scan Filters: informix.product.product_number IN (1 , 2 )

  (1) Index Name: (Auto Index)
      Index Keys: product_code
```

System Generated Indexes

- Constraints will create a system generated index if one does not already exist
- System generated index names begin with a space

System Generated Indexes



System Generated

User Created

Best Practices for Creating Constraints

- Create the table
- Load the data
- Create the indexes that the constraints will require
- Alter table to add constraints, it will use the existing indexes

Difference of Unique Index vs Unique Constraint

- For Logged tables, unique indexes are checked on every row
- For Logged tables, unique constraints are checked at the end of the SQL statement or on commit

Foreign Key Constraints Without an Index

- Very useful for large Fact Tables which reference small Dimension Tables where Foreign Key Constraint indexes create additional overhead
- Create the indexes that the constraints will require with the index Disabled
- Alter table to add constraints; it will use the existing Disabled index

Foreign Key Constraints Without an Index

```
alter table bills add constraint
        ( foreign key ( state) references state ( state )
        constraint fk_bills_state index disabled )
```

Best Practices for Loading Large Tables With Indexes

- Disable the indexes
- Set the table to RAW
- Load the data
- Set the table to STANDARD
- Enable the indexes and this will rebuild the indexes

Clustered Indexes

- Perform one-time sorts that reorder physical data rows to match index order
- Requires enough free space to build a copy of the table during the process
- Locks table during rebuild process
- Syntax:
  - alter index index_name to cluster
- Maintenance operation

Composite Indexes

- Order of columns is important
  - Example: Which indexes are better?
    - (first_name, last_name, company)
    - (company, first_name, last_name)
    - (company, last_name, first_name)
    - (company)
    - (first_name, last_name)
- Avoid unneeded redundant indexes

Composite Indexes

- Order of columns is important
  - Example: Which indexes are better?
    - ~~(first_name, last_name, company)~~
    - ~~(company, first_name, last_name~~)
    - (company, last_name, first_name)
    - ~~(company)~~
    - ~~(first_name, last_name)~~
    - Add index (last_name, company)

Index Fill Factor

- What percent of an index page is filled when the index is first created?
- Default is 90
- Use 100 for index pages that never grow
- Use 50 for high-growth tables
  - Create index index_name on table X (col1) fillfactor 50;

What is a Functional Index?

- A functional index stores the result of a function on a column in an index
  - Create index idx_zone on buildings (area(length,width));
  - Create index idx_last_name_upper on customers (toupper (last_name));

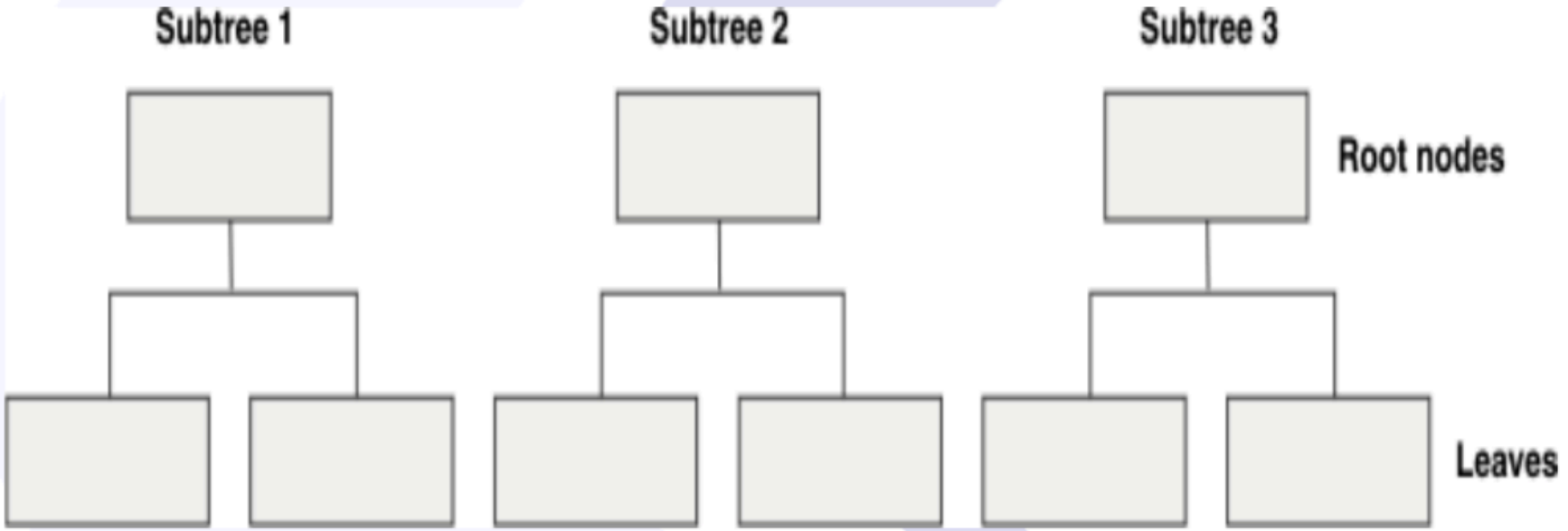
Functional Index

```
drop function toupper ( varchar(255) ) ;

create function toupper ( name varchar(255) ) returns varchar (255)
        with ( not variant );
        return upper (name );
end function;

select toupper(state_name)  from state;

create index idx_state_name_upper on state ( toupper ( state_name ) );
~
~
```

What is a Forest of Trees Index?

- A forest of trees index is like a B-tree index, but it has multiple root nodes and potentially fewer levels
- Multiple root nodes can alleviate root node contention, because more concurrent users can access the index
- A forest of trees index can also improve the performance of a query by reducing the number of levels involved in read operations

What is a Forest of Trees Index?



What is the Best DBspace Page Size for an Index?

- For large tables:
  - Create a DBspace with 16 KB page size
  - Create a separate BUFFER pool for the 16 KB DBspaces

Index Partitioning

- Locate indexes in separate dbspace from data
- Partition indexes in the same way the table is partitioned
  - by expression, list, or interval range
  - this enables detaching table partitions when the indexes match the table
- Can partition indexes using a different expression than the data, but…

Automatic Partition by Interval with Partitioned Index

lester — ssh informix@tiger8 — 120×40

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

Attached vs Detached Indexes

- Default in version 7.X – Attached Indexes
  - index pages are stored within the same tablespace as data pages
- Default in version 9.X and greater - Detached Indexes
  - index pages are stored in separate tablespace from the data pages

New Index Features in Informix 14

- When a varchar column length is being increased, the index is not rebuilt
- Rename system generated indexes
  - rename index 104_8 to idx_bills_pk;
- Drop constraint and maintain index
- Rename system generated constraints
  - rename constraint xyz to mytab_pk_const;

How to Speed up Index Creation?

- Creating indexes requires CPU threads, memory, and temporary space
- Use PDQ Priority
- Have at least three temporary dbspaces with enough space to rebuild the index
- Use PSORT_NPROCS and PSORT_DBTEMP

How Do You Maintain Indexes for Peak Efficiency?

- To rebuild Indexes
  - Drop Index
  - Create the Index
  - See section on speeding up index builds
- **DO NOT use kill 9 on a user session, or the server, as this may leave indexes in inconsistent state**

BTSCANNER

- The B-Tree Scanner threads remove deleted index entries and rebalance the index nodes
- ONCONFIG Value
  - BTSCANNER num=1,threshold=5000,rangesize=-1,alice=6,compression=default
  - Num is number of threads to run
  - Threshold is minimum number of deleted keys to prioritize for cleaning
  - Rangesize is mode of scanning where -1 is alice mode and 100 is leaf scan mode
  - Alice controls index cleaning where
    - 0 = off
    - 1 = Uses 8 bytes
    - 2 = Use 16 bytes
    - 3-12 = Sets the start memory for cleaning and may be automatically adjusted (default is 6)
  - Compression is the level that partially-used pages may be merged
    - Low
    - Med or default
    - High

Starting Additional B-Tree Scanner Threads

- Onmode command to add B-Tree Scanners:

Onmode -C {start <count>|stop <count>|threshold <size>|
duration <seconds>|rangesize <size>|alice <mode>|
compression <low|med|high|default>}

  - onmode -C start 1
  - onmode -C stop 1

```
informix@tiger1:~ train1 > onstat -g ath | grep btscanner
 22       4b7cb178         4a6ae8e8         1    sleeping secs: 57       8cpu         btscanner_0
informix@tiger1:~ train1 > onmode -C start 1
informix@tiger1:~ train1 > onstat -g ath | grep btscanner
 22       4b7cb178         4a6ae8e8         1    sleeping secs: 46       8cpu         btscanner_0
 12728    4fba5a00         4a6b5368         1    cond wait  btc_sort     1cpu         btscanner_1
informix@tiger1:~ train1 > onmode -C stop 1
informix@tiger1:~ train1 > onstat -g ath | grep btscanner
 22       4b7cb178         4a6ae8e8         1    sleeping secs: 35       8cpu         btscanner_0
```

Onstat Commands to Monitor B-Tree Scanners

Onstat Command	Description
onstat –C	Print profile information for the system and scanner threads
onstat –C prof	Print profile information for the system and scanner threads
onstat –C hot	Print hot list index keys
onstat –C part	Print all partitions with index statistics
onstat –C clean	Print information about all partitions cleaned and need to be
onstat –C range	Print savings in pages processed with range scanning
onstat –C map	Print current alice bitmap for all indexes being cleaned
onstat –C alice	Print efficiency of alice cleaning method
onstat –C all	Print all onstat -C options

Onstat –C
to Monitor B-Tree Scanners

```
IBM Informix Dynamic Server Version 14.10.FC1 -- On-Line -- Up 15 days 23:56:11

B-tree Scanner Information
Profile
=======
Active Threads                              1
Global Commands                       2000000   Building hot list
Number of partition scans               15967
Main Block                      0x000000004b063d18
BTC Admin                       0x000000004a6ae8e8


BTS info             id   Prio    Partnum      Key     Cmd
0x000000004b0719d8    0   High   0x00000000     0      40  Yield N
    Number of leaves pages scanned                      1777
    Number of leaves with deleted items                  415
    Time spent cleaning (sec)                              0
    Number of index compresses                           259
    Number of deleted items                            39035
    Number of index range scans                            0
    Number of index leaf scans                            29
    Number of index alice scans                           47
```

Onchecks for Indexes

- Oncheck options to check indexes
  - i - table indexes database[:[owner.]table[#index]]
  - I - table indexes and rowids in index database[:[owner.]table[#index]]
- Examples:
  - oncheck –ci database:table#index_name
  - oncheck –cI database:table#index_name
- Best option check all indexes and data
  - oncheck –cDI database:table

New Oncheck to Show Index Last Used – oncheck -pT

```
Index Usage Report for index  104_8 on benchmark2:informix.bills

                  Average    Average  Average
    Level    Total No. Keys Free Bytes Del Keys
    ----- -------- -------- ---------- --------
        1        1       94        896
        2       94       83       1016
        3     7861      153         18        0
    ----- -------- -------- ---------- --------
    Total     7956      153         29        0

                Index idx_bills_customer_number fragment partition datadbs in DBspace datadbs

    Physical Address                3:571
    Creation date                   08/16/2019 14:14:35
    TBLspace Flags                  801        Page Locking
                                               TBLspace use 4 bit bit-maps
    Maximum row size                1266
    Number of special columns       0
    Number of keys                  1
    Number of extents               24
    Current serial value            1
    Current SERIAL8 value           1
    Current BIGSERIAL value         1
    Current REFID value             1
    Pagesize (k)                    2
    First extent size               4
    Next extent size                512
    Number of pages allocated       4096
    Number of pages used            3678
    Number of data pages            0
    Number of rows                  0
    Partition partnum               3145800
    Partition lockid                3145798
    Last Lookup/Scan                Fri Aug 16 14:37:09 2019
```

Index Last Used

How to Monitor Index Usage

Compare reads and writes on an index…fewer reads indicates the index may not be needed unless it is a constraint

My old script – does not handle Indexes with the same names in different databases

```
Select      a.tabname,
            b.idxname,
            bufreads,
            bufwrites,
            case
                        when bufwrites = 0 then bu      ds
                        when bufreads = 0 then 0
                        else ( bufreads /bufwrites )
            end ratio
from systables a, sysindexes b,  outer sysmaster:sysptprof p
where a.tabid = b.tabid
and   p.tabname = b.idxname
and a.tabid > 99;
```

New - How to Monitor Index Usage

Compare reads and writes on an index by partition …fewer reads indicates the index may not be needed unless it is a constraint

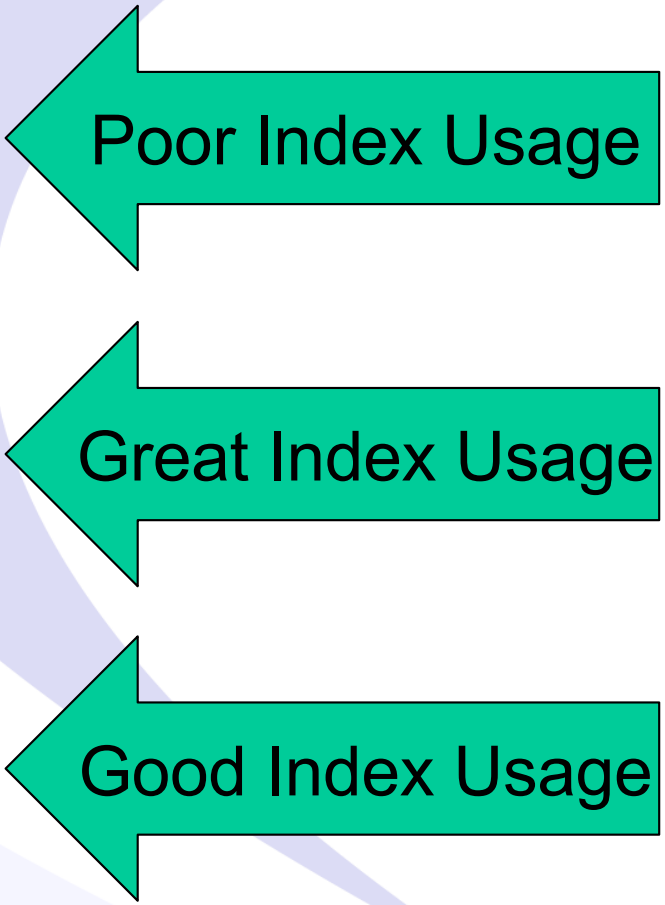
```
select
        t.tabname,
        i.indexname,
        bufreads,
        bufwrites,
        case
                when bufwrites = 0 then bufreads
                when bufreads = 0 then 0
                else ( bufreads /bufwrites )
        end ratio
from    systables t, sysfragments i,  outer sysmaster:sysptprof p
where   t.tabid = i.tabid
and     i.fragtype = "I"
and     i.partn = p.partnum
and     t.tabid > 99;
```

New - How to Monitor Index Usage

```
-- ###############################################################################
-- ## Module: @(#)index_usage2.sql      2.0     Date: 08/25/2019
-- ## Author: Lester Knutsen  Email: lester@advancedatatools.com
-- ##         Advanced DataTools Corporation
-- ###############################################################################

select
        t.tabname,
        i.indexname,
        bufreads,
        bufwrites,
        case
                when bufwrites = 0 then bufreads
                when bufreads = 0 then 0
                else ( bufreads /bufwrites )
        end ratio
from    systables t, sysfragments i,  outer sysmaster:sysptprof p
where   t.tabid = i.tabid
and     i.fragtype = "I"
and     i.partn = p.partnum
and     t.tabid > 99;
```

New - How to Monitor Index Usage

```
tabname     state
indexname   idx_state_1
bufreads    15
bufwrites   11
ratio       1.36363636363636

tabname     zip
indexname   idx_zip_1
bufreads    630617
bufwrites   931
ratio       677.354457572503

tabname     benchmark
indexname   idx_benchmark_1
bufreads    214154
bufwrites   4614
ratio       46.4139575205895
```



Questions?

Send follow-up questions to
Lester@advancedatatools.com

IIUG World 2019

More information at

http://www.iiug.org

Advanced DataTools Sessions at IIUG World 2019

Mon 9/23	10:15 AM	C01. Getting Going With Informix Connection Manager	Thomas Beebe
	11:30 AM	A02. Stories from Database Support Customers	Mike Walker
	11:30 AM	D02. Configuring and using the Informix Warehouse Accelerator	Art Kagel
	1:30 PM	B03. Do Stats Better	Art Kagel
Tue 9/24	11:30 AM	B07. Do Storage Better	Art Kagel
	2:45 PM	C09. Setting up SSL for Informix	Thomas Beebe
Wed 9/25	9:30 AM	C11. Migrating your Informix Instance	Mike Walker
	10:15 AM	A12. Exploring the Sysmaster - my new stuff	Lester Knutsen
	2:45 PM	B15. Set up a Raspberry Pi	Mike Walker

Free Informix Webcasts

from the IBM Informix Champions

- **SQL Explain - Using the SQL Optimizer Query Explain Plan, by Lester Knutsen**
  - **Thursday, October 31, 2019 at 2:00pm EDT**

- **Update Statistics - Best Practices for Informix DBAs, by Lester Knutsen**
  - **Thursday, November 21, 2019 at 2:00pm EDT**

Registration and more information:
https://advancedatatools.com/Informix/NextWebcast.html

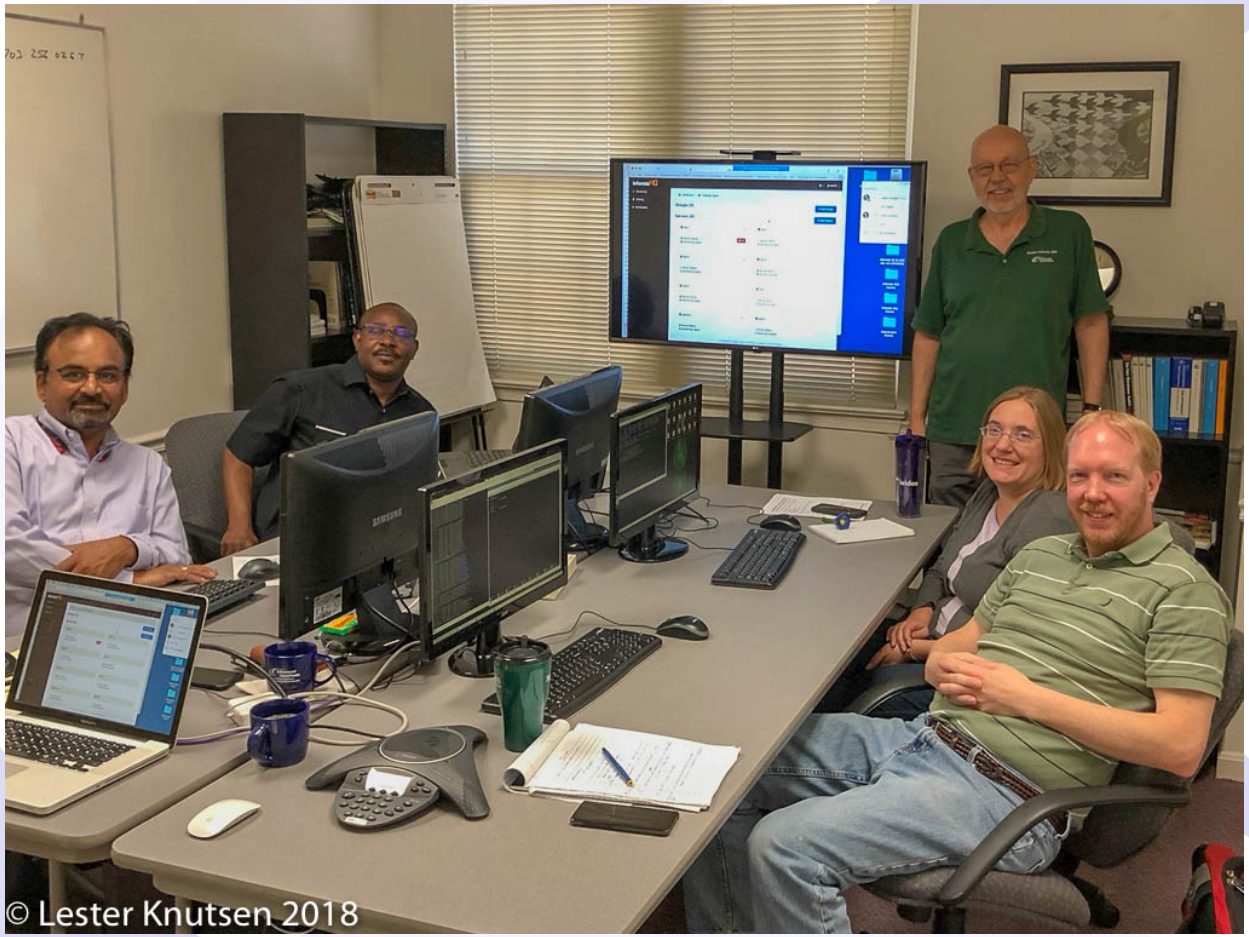
Informix Training
Updated for Informix 14.10

Attend classes online on the web, or in person at our training center in Virginia. All you need is a web browser to connect to our WebEx training system and an SSH client (like Putty) to connect to our training lab for hands-on exercises. Each student uses an 8-core Linux server, with 16GB RAM, SSD drives with Informix 12, and several large databases for benchmark exercises.

- **March 11-14, 2019 - Advanced Informix Perfo** Completed **g**
- **April 22-25, 2019 - Informix for Database Adm** Completed
  - This course is for new database administrators, programmers, and technical support personnel who will be setting up, managing, and tuning IBM Informix databases.
- **October 7-10, 2019 - Informix for Database Administrators**
  - This course is for new database administrators, programmers, and technical support personnel who will be setting up, managing, and tuning IBM Informix databases.

More information and registration at:
http://www.advancedatatools.com/Training/InformixTraining.html

Informix 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