Advanced DataTools Webcast

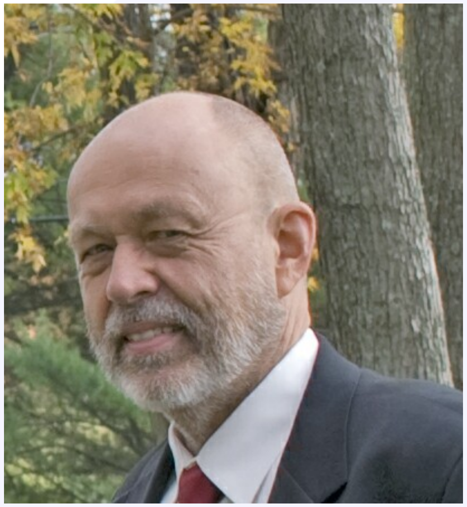
from the IBM Informix Champions

Using the Informix SQL Optimizer Query Explain Plan

by Lester Knutsen

Thursday, October 31, 2019 at 2:00pm EDT

Lester Knutsen

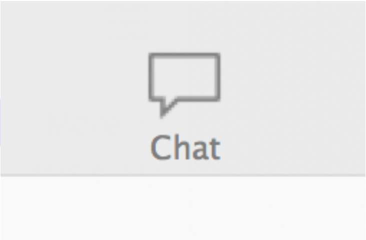


Lester Knutsen is President of Advanced DataTools Corporation, and has been building large data warehouse and business systems using Informix Database software since 1983. Lester focuses on large database performance tuning, training, and consulting. Lester is a member of the IBM Gold Consultant program 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.



Understanding the Informix SQL Optimizer Using Set Explain

For Every SQL Statement

- The Informix Server:
  - Checks Syntax
  - Checks Permissions
  - Optimizes the SQL statement to determine the best access method
  - Develops a SQL plan
  - Executes the SQL statement

Informix Query Optimizer

- The query **optimizer** attempts to determine the most efficient way to execute a SQL statement
- Examines every possible method to implement the query and **selects the least costly method**
- NOT rule based; does not follow the order of tables or fields
- Dynamic, so when data changes the same SQL can select a better path

For Every SQL Statement

- Optimizes the SQL statement to determine the best access method
  - – Which table to read first….
  - – Which index to use first…
  - – Which filter to use first…

Factors Effecting the Optimizer

- Number of rows in each table
- Number of pages used
- How unique are the columns
- What are the indexes
- How many levels are the indexes
- The distribution of data

To See the SQL Query Plan

- Set EXPLAIN ON
  - Display the query plan that the optimizer chooses, and execute the query.
- Set EXPLAIN ON FILE TO “filename”
  - Save the query plan in a specific file
- Set EXPLAIN ON AVOID_EXECUTE
  - Display the query plan that the optimizer chooses, but do not execute the query.
- onmode -Y sid
  - Display the query plan that the optimizer chooses for a sid.
    - EXECUTE FUNCTION task("onmode","Y","101","1");
    - onmode –Y 101 1 /tmp/users_sqexplain.out

SQL EXPLAIN Output

- The SQL/SELECT statement for the query
- Estimate of the query cost (in units) the optimizer uses to compare plans.
  - Units represent a relative time for query execution, with each unit assumed to be roughly equivalent to a disk access.
  - The optimizer chooses the query plan with the lowest estimated cost for its execution.
- An estimate for the number of rows that the query is expected to produce
- The order to access the tables during execution
- The access plan by which the database server reads each table

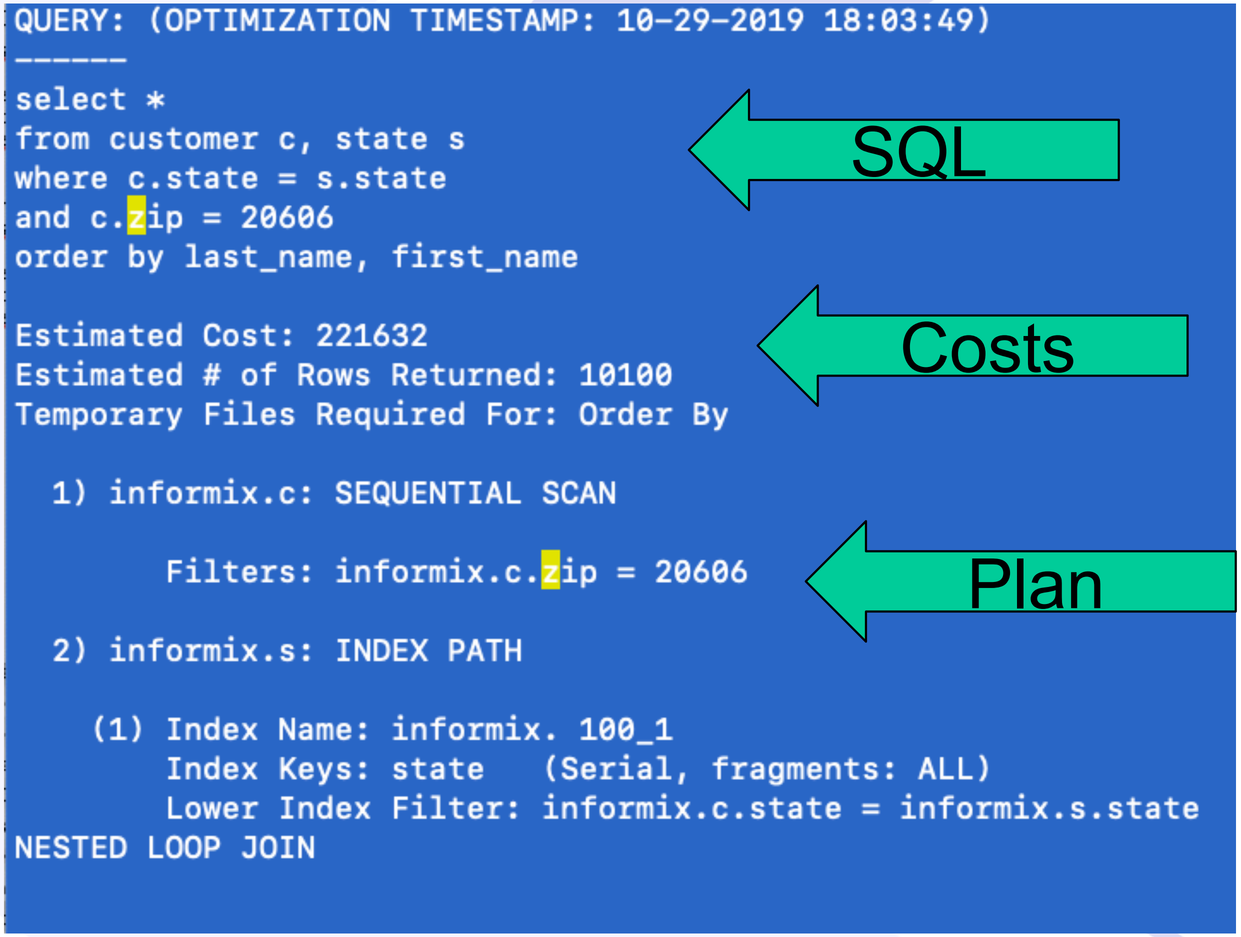
Explain Output – Access Plan

- SEQUENTIAL SCAN
  - Reads all rows in sequence
- INDEX PATH
  - Scans one or more indexes
- AUTOINDEX PATH
  - Creates a temporary index
- REMOTE PATH
  - Accesses another database (distributed query)
    - The table column(s) that serve as a filter, if any, and whether the filtering occurs through an index
    - The join plan for each pair of tables

Explain Output – Join Plan

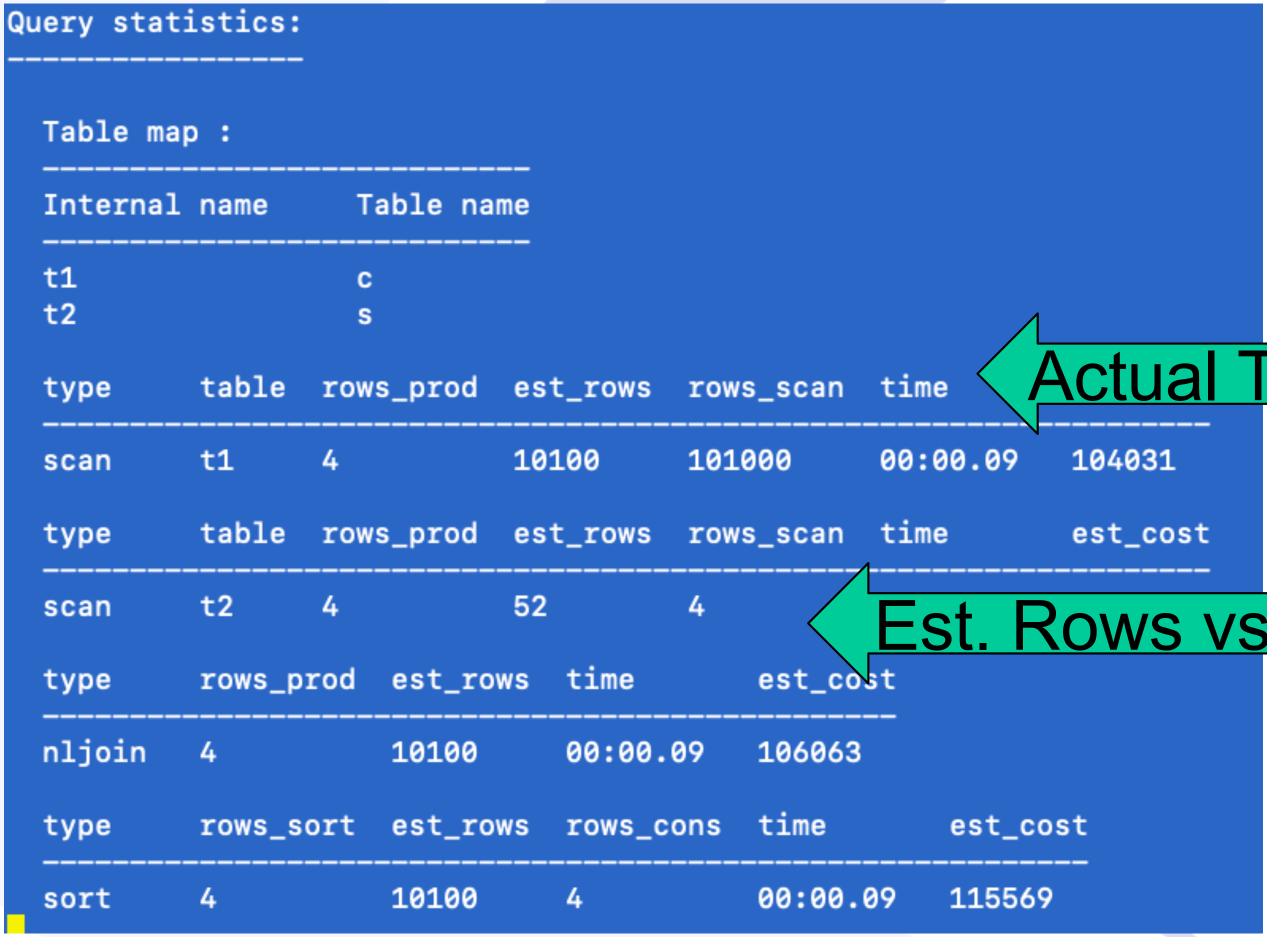
- DYNAMIC HASH
  - Use a hash join on the preceding join-table pair. The output includes a list of the filters used to join the tables. If DYNAMIC HASH JOIN is followed by (Build Outer) in the output, the build phase occurs on the first table. Otherwise, the build occurs on the second table, preceding the DYNAMIC HASH JOIN.
- NESTED LOOP
  - Use a hash join on the preceding join-table pair. The output includes a list of the filters used to join the tables. The optimizer lists the outer table first for each join pair.

Sqexplain.out – Part 1
Query Plan



```
QUERY: (OPTIMIZATION TIMESTAMP: 10-29-2019 18:03:49)
------
select *
from customer c, state s
where c.state = s.state
and c.zip = 20606
order by last_name, first_name

Estimated Cost: 221632
Estimated # of Rows Returned: 10100
Temporary Files Required For: Order By

  1) informix.c: SEQUENTIAL SCAN

        Filters: informix.c.zip = 20606

  2) informix.s: INDEX PATH

    (1) Index Name: informix. 100_1
        Index Keys: state   (Serial, fragments: ALL)
        Lower Index Filter: informix.c.state = informix.s.state
NESTED LOOP JOIN
```

Sqexplain.out – Part 2
Query Statistics

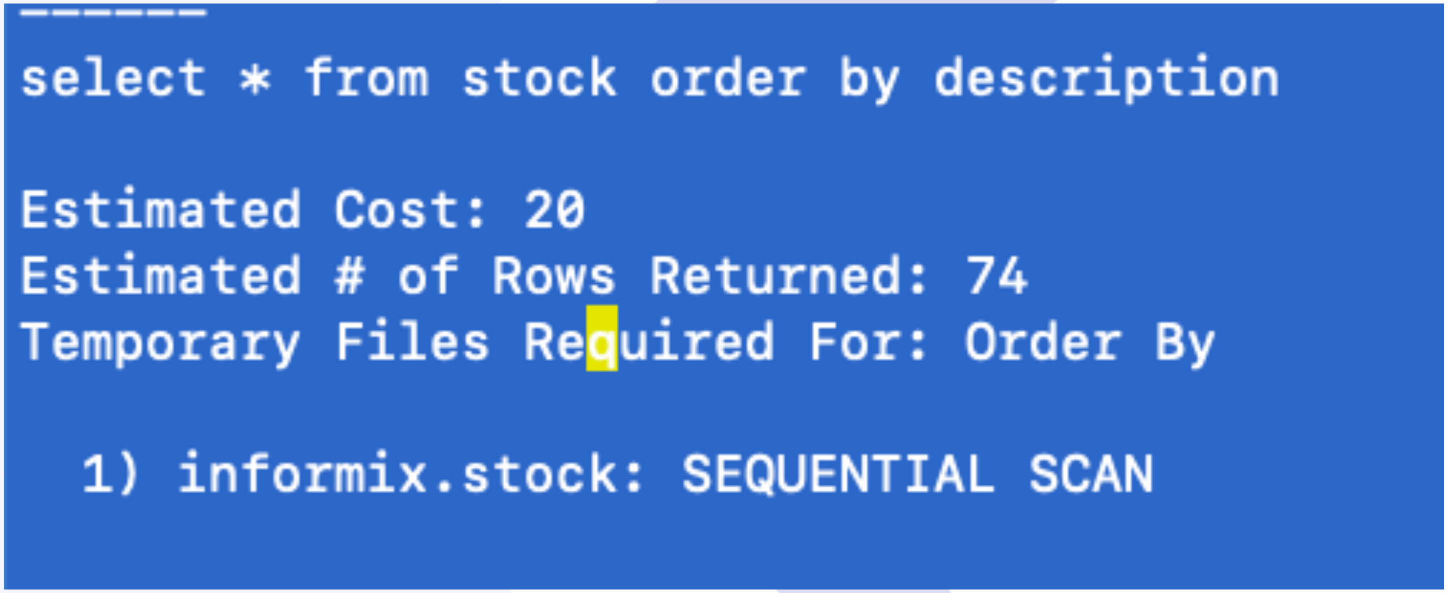


```
Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  t1                c
  t2                s

  type     table  rows_prod  est_rows  rows_scan  time
  -------------------------------------------------------------------
  scan     t1     4          10100     101000     00:00.09   104031

  type     table  rows_prod  est_rows  rows_scan  time       est_cost
  -------------------------------------------------------------------
  scan     t2     4          52        4

  type     rows_prod  est_rows  time       est_cost
  -------------------------------------------------
  nljoin   4          10100     00:00.09   106063

  type     rows_sort  est_rows  rows_cons  time       est_cost
  ------------------------------------------------------------
  sort     4          10100     4          00:00.09   115569
```

Actual Time

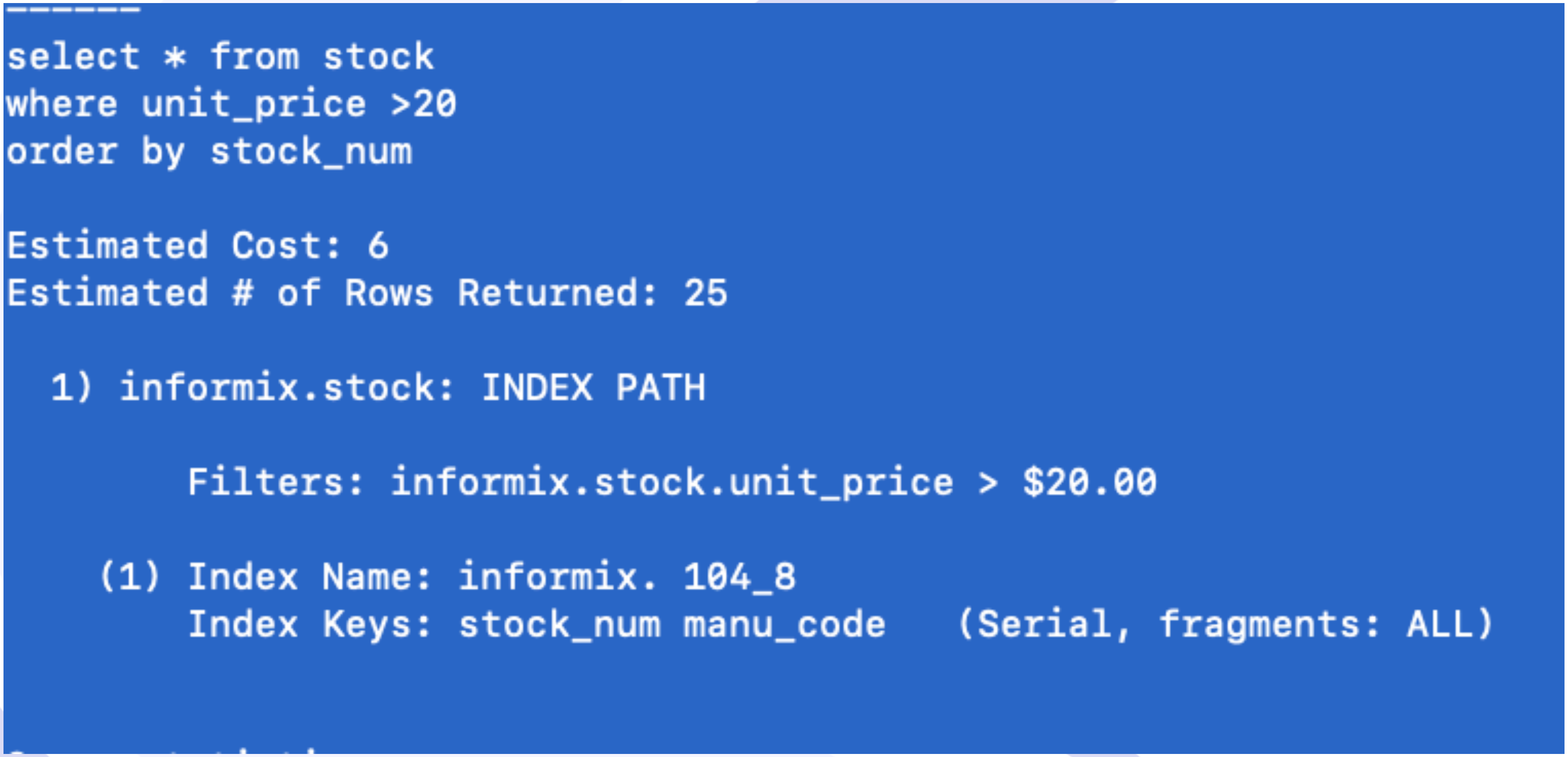
Est. Rows vs Actual

Set Explain: Example 1
Sequential Scan and Sort



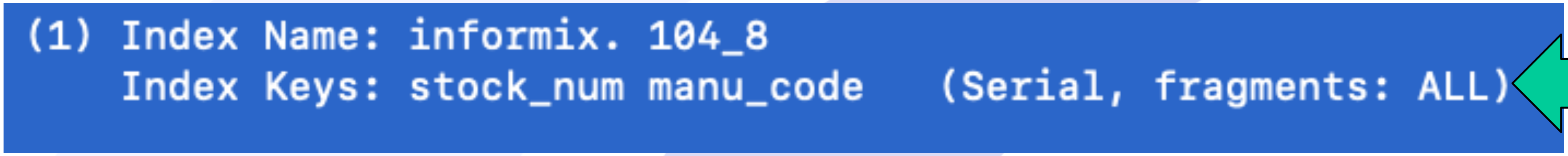
```
-------
select * from stock order by description

Estimated Cost: 20
Estimated # of Rows Returned: 74
Temporary Files Required For: Order By

  1) informix.stock: SEQUENTIAL SCAN
```

Set Explain: Example 2
Index Path for Sort



```
-------
select * from stock
where unit_price >20
order by stock_num

Estimated Cost: 6
Estimated # of Rows Returned: 25

  1) informix.stock: INDEX PATH

        Filters: informix.stock.unit_price > $20.00

    (1) Index Name: informix. 104_8
        Index Keys: stock_num manu_code   (Serial, fragments: ALL)
```

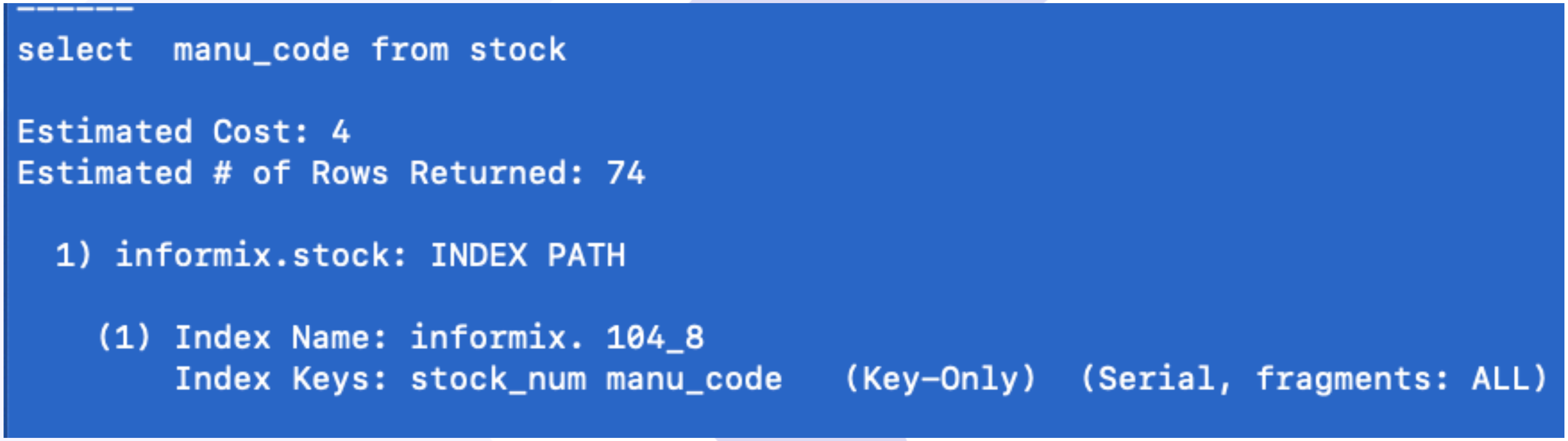
Index Path



```
(1) Index Name: informix. 104_8
    Index Keys: stock_num manu_code   (Serial, fragments: ALL)
```

- Serial – the server reads one index fragment at a time
- Parallel – the server reads all index fragments at the same time
- Fragments
  - Fragments: ALL – the server reads all index fragments
  - Fragments: [name list] – names of the index fragments the server will read

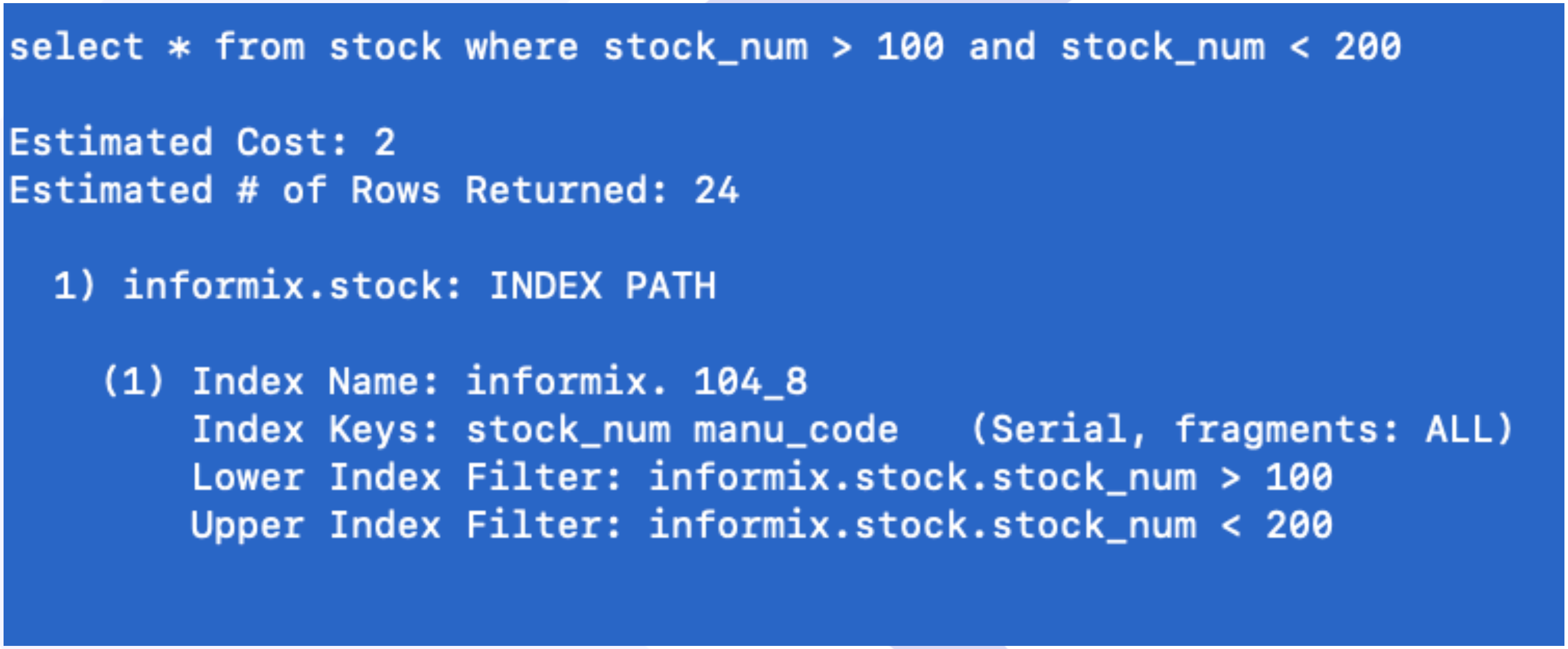
Set Explain: Example 3
Key-Only Path



```
------
select  manu_code from stock

Estimated Cost: 4
Estimated # of Rows Returned: 74

  1) informix.stock: INDEX PATH

    (1) Index Name: informix. 104_8
        Index Keys: stock_num manu_code   (Key-Only)  (Serial, fragments: ALL)
```

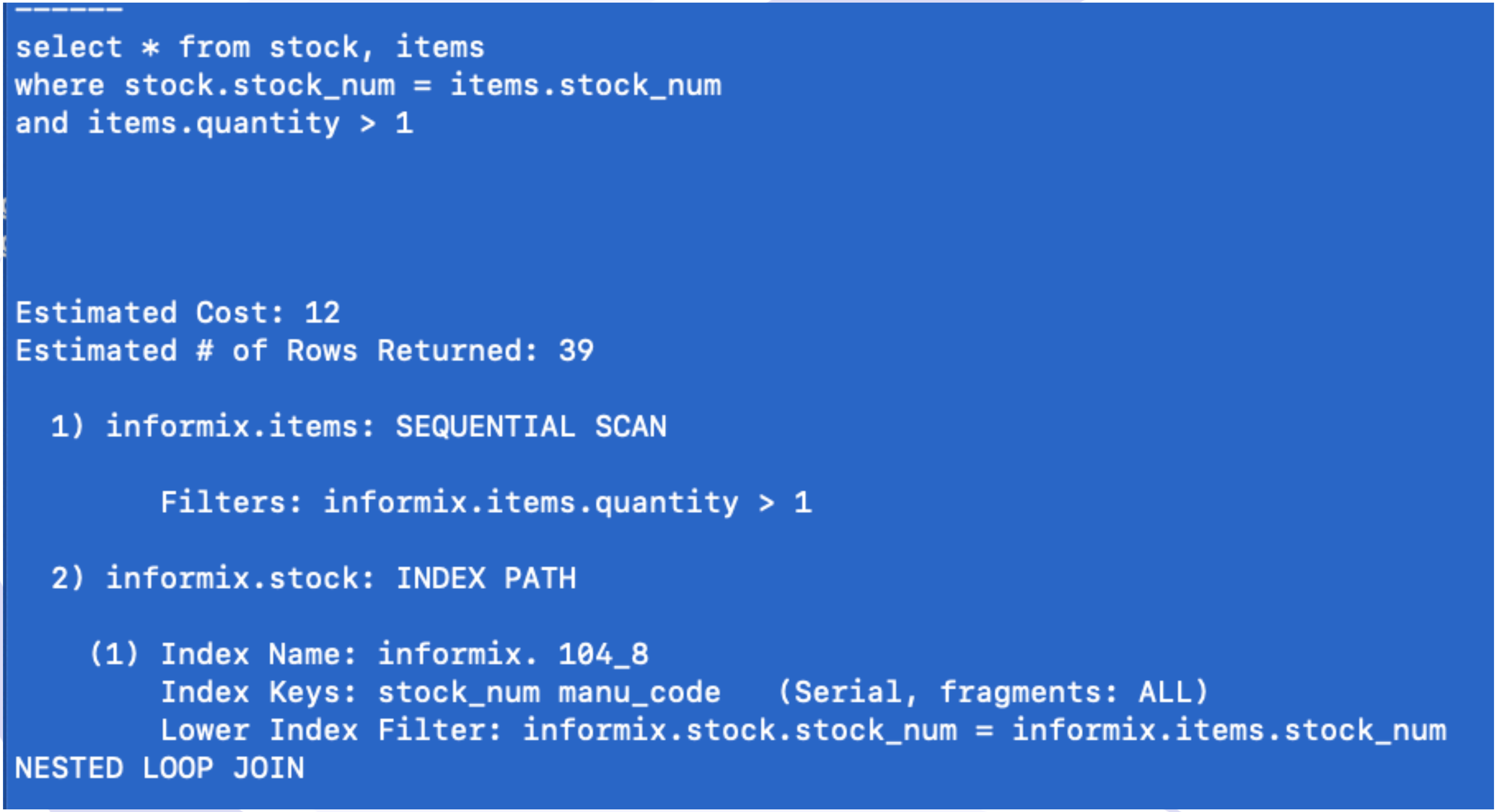
- The Index contains all the information requested and only the Index is read
- No need to read the table
- **Very Efficient**

Set Explain: Example 4
Where Clause Filters



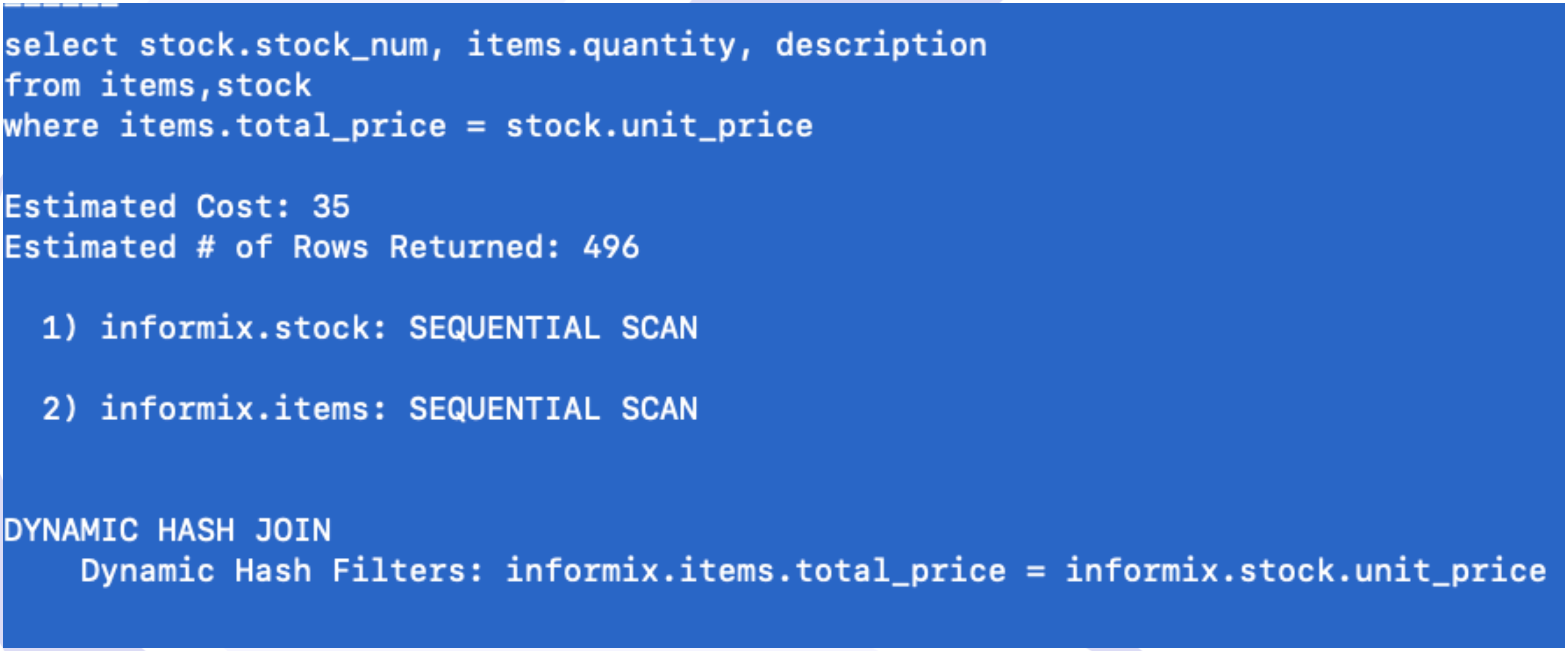
```
select * from stock where stock_num > 100 and stock_num < 200

Estimated Cost: 2
Estimated # of Rows Returned: 24

  1) informix.stock: INDEX PATH

    (1) Index Name: informix. 104_8
        Index Keys: stock_num manu_code   (Serial, fragments: ALL)
        Lower Index Filter: informix.stock.stock_num > 100
        Upper Index Filter: informix.stock.stock_num < 200
```

Set Explain: Example 5
Table Joins – Nested Loop



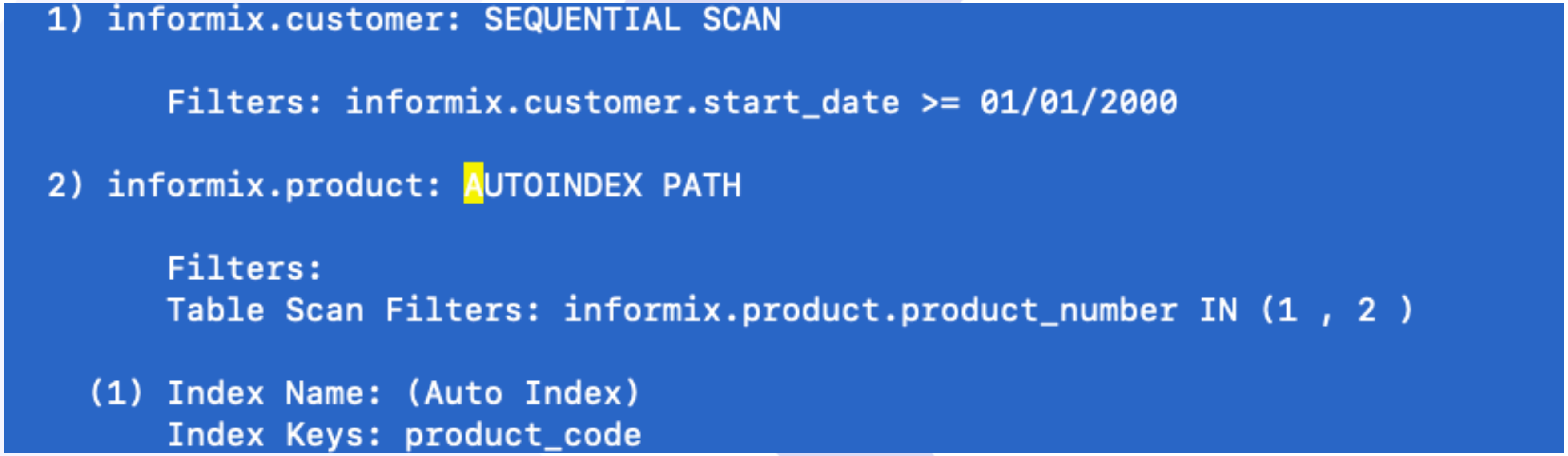
```
------
select * from stock, items
where stock.stock_num = items.stock_num
and items.quantity > 1




Estimated Cost: 12
Estimated # of Rows Returned: 39

  1) informix.items: SEQUENTIAL SCAN

        Filters: informix.items.quantity > 1

  2) informix.stock: INDEX PATH

    (1) Index Name: informix. 104_8
        Index Keys: stock_num manu_code   (Serial, fragments: ALL)
        Lower Index Filter: informix.stock.stock_num = informix.items.stock_num
NESTED LOOP JOIN
```

Set Explain: Example 6
Table Join – Dynamic Hash



```
select stock.stock_num, items.quantity, description
from items,stock
where items.total_price = stock.unit_price

Estimated Cost: 35
Estimated # of Rows Returned: 496

  1) informix.stock: SEQUENTIAL SCAN

  2) informix.items: SEQUENTIAL SCAN


DYNAMIC HASH JOIN
    Dynamic Hash Filters: informix.items.total_price = informix.stock.unit_price
```

Set Explain: Example 7
Auto Index



```
  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
```

Influence the Query Optimizer

- Update Statistics
- Query Directives
- Using parentheses ( )
- Indexes
- Selectivity of columns used in filters
- Many ways to write the same SQL

Optimizer Performance

- Query Flattening - turns a subquery (inner select statement) into a join with a table in the outer query to improve performance
- Query re-write – re-writes a query to improve performance

Optimizer Directives

- Hints to the Optimizer on how to execute the SQL
- May be used to fix poor statistics
- May produce slower performance

Types of Directives

- Access-Method Directives
- Join-Order Directives
- Join-Method Directives
- Star-Join Directives
- Optimization-Goal Directives
- Explain-Mode Directives
- Statement Cache Directives

Syntax of Directives

- Directives are placed in SQL comments
- --+ Directive
- {+ Directive }
- /*+ Directive */
- Example
  - SELECT {+ INDEX (emp idx_dept_no) }

Access-Method Directives

- AVOID_FULL
- AVOID_INDEX
- AVOID_INDEX_SJ
- FULL
- INDEX
- INDEX_ALL
- MULTI_INDEX
- INDEX_SJ

Examples

- Q01 – Sequential scan
- Q02 – Sort
- Q03 – Filter
- Q04 – Two table join
- Q05 – Two table join and two filters
- Q06 – Two filters (which is better?)
- Q07 – Two filters with two indexes (which is better?)
- Q08 – Query Directive
- Q09 – Three table join
- Q10 – Outer join

More Examples

- Qa01 – Compare Informix Joins with ANSI Joins
- Qa02 – Compare Informix Joins with ANSI Joins
- Qa03 – Informix Self Join
- Qa04 – Compare Or with Union statement
- Qa05 – Compare temp with derived table
- Qa06 – Compare matches with ANSI like
- Qa07 – Case statement
- Qa08 – Compare queries' performance – same results
- Qa09 – Compare performance of Update statements

Questions?

Send follow-up questions to
Lester@advancedatatools.com

Free Informix Webcasts

from the IBM Informix Champions

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

- **Coming in 2020 – Informix Tutorials Webcast Series**
  - **One FREE training tutorial a month**

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

Informix Training
Updated for Informix 14.10

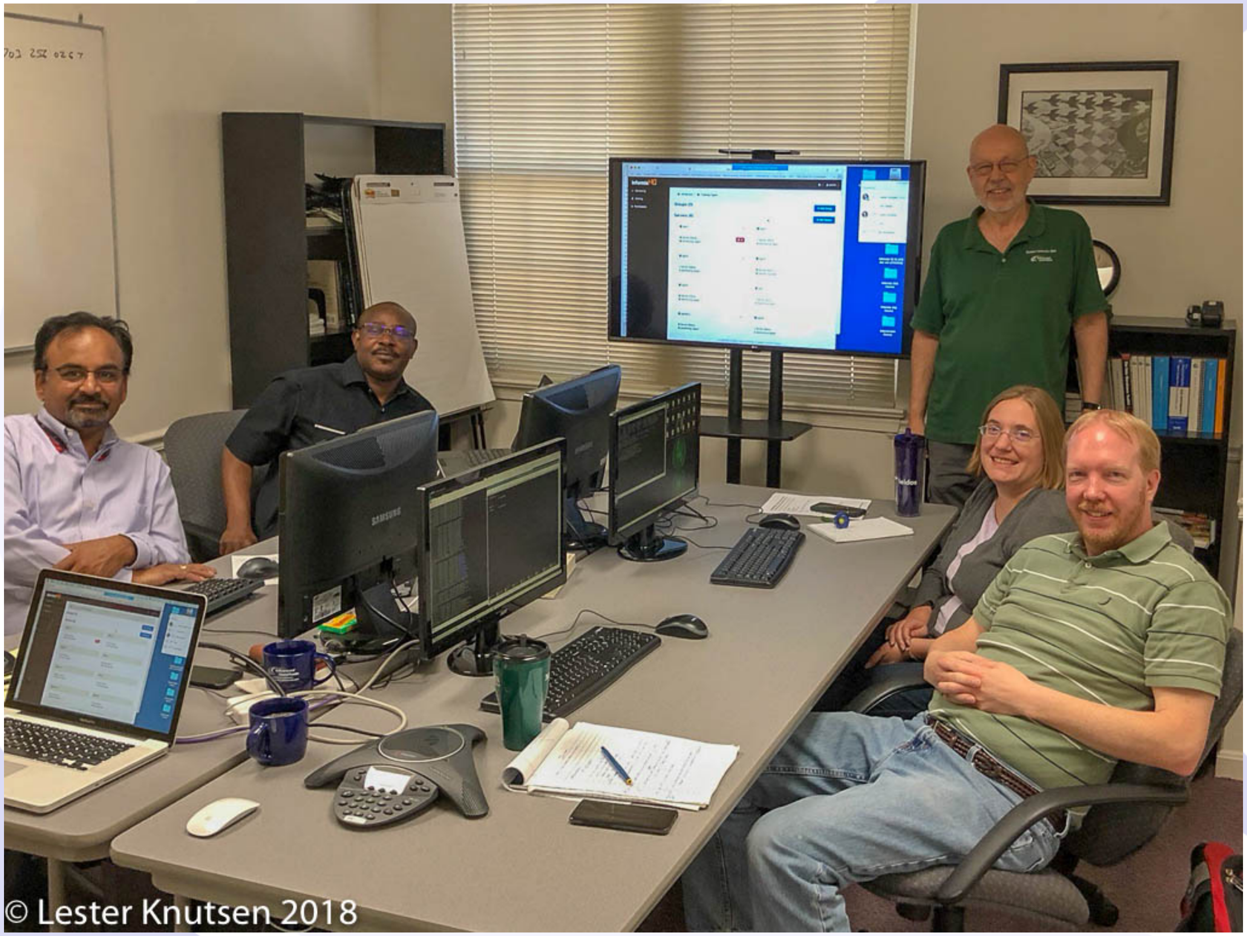
Attend classes online on the web or in person at our training center in Virginia. All you need is a web browser to connect to our WebEx training system, and an SSH client (like Putty) to connect to our training lab for hands-on exercises. Each student uses an 8-core Linux server, with 16GB RAM, SSD drives with Informix 14, and several large databases for benchmark exercises.

Informix Training in 2020

- **May 18-21, 2020 - Informix for Database Administrators**
- **July 6-9, 2020 - Advanced Informix Performance Tuning**
- **October 5-8, 2020 - Informix for Database Administrators**

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

Informix 14 Training



Each student in class will have a server running Informix 14.10 with:

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

Class size is limited to 8 students.

Attend online or in person!

Advanced DataTools

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, 3 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!

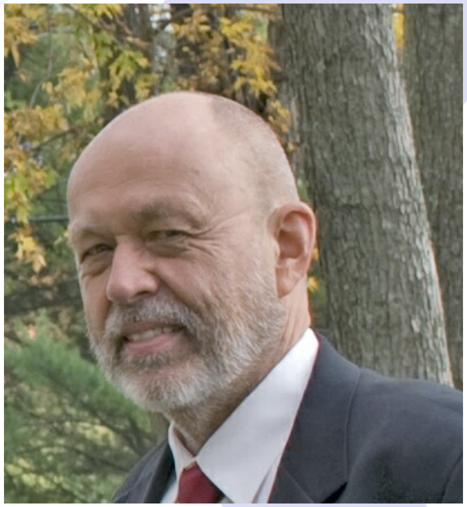
- ***Informix Remote DBA Support Monitoring***
- ***Informix Performance Tuning***
- ***Informix Training***
- ***Informix Consulting***
- ***Informix Development***

Free Informix Performance Tuning Webcast replays at:
http://advancedatatools.com/Informix/Webcasts.html
Email: info@advancedatatools.com
Web: http://www.advancedatatools.com

IBM Business Partner

Thank You

Advanced DataTools Corporation



For more information:

Lester@advancedatatools.com

http://www.advancedatatools.com