# Interviewing the DB2 LUW Optimizer


***Ember Crooks***

*XTIVIA (http://www.virtual-dba.com/, http://www.xtivia.com/)*

*Blog: http://db2commerce.com*

*Twitter: @ember_crooks*

# Objectives

- Understand how to gather explain information and the advantages and disadvantages of the various methods.
- Read an explain plan
- Using the design advisor in conventional and unusual ways
- Collect section actuals
- Understand some SQL tuning strategies that come out of explain data

# What is the DB2 Optimizer?

- SQL is a language that describes the data to return, but not where that data is or how to access it
- The DB2 optimizer:
  - Makes cost-based decisions on how to best access the data needed to fulfill a query
  - Rewrites queries in some ways that make sense
  - Relies on statistics gathered about data to make the best decisions
  - Knows about the DB2 environment and uses environmental advantages and restrictions in its decisions
  - Can be limited and influenced by parameters, optimization profiles, and hints
  - Generates access plans (sections) that are stored in the package cache
  - Is sometimes referred to as the SQL compiler

# Optimization Time vs. Execution Time

- For more complex queries, more optimization time can lead to lower execution time
- For simple queries, this may not be true
- DB CFG parameter DFT_QUERYOPT sets the query optimization class
- CURRENT QUERY OPTIMIZATION special register can be used for this on a session basis
- Often the default of 5 works just fine

# Parameter Markers

- Actual statement:

```
select * from users where
users_id = 2 and update time <
current timestamp – 7 days
```

- Statement prepared by application and stored in the Package Cache:

```
select * from users where
users_id = ? and update time <
current timestamp – ? Days
```

- Values 2 and 7 are passed at execution time, not preparation time

*Parameter markers allow actual values to be filled in at execution time*

*The same statement with different values uses the same plan/section from the package cache*

*If parameter markers are not appropriately used, every statement will be treated as different and have a different entry in the package cache*

*The application chooses whether or not to use parameter markers*

*Parameter markers are not always best if the data is very unevenly distributed*



XTIVIA

# How to Gather and Display Explain Information

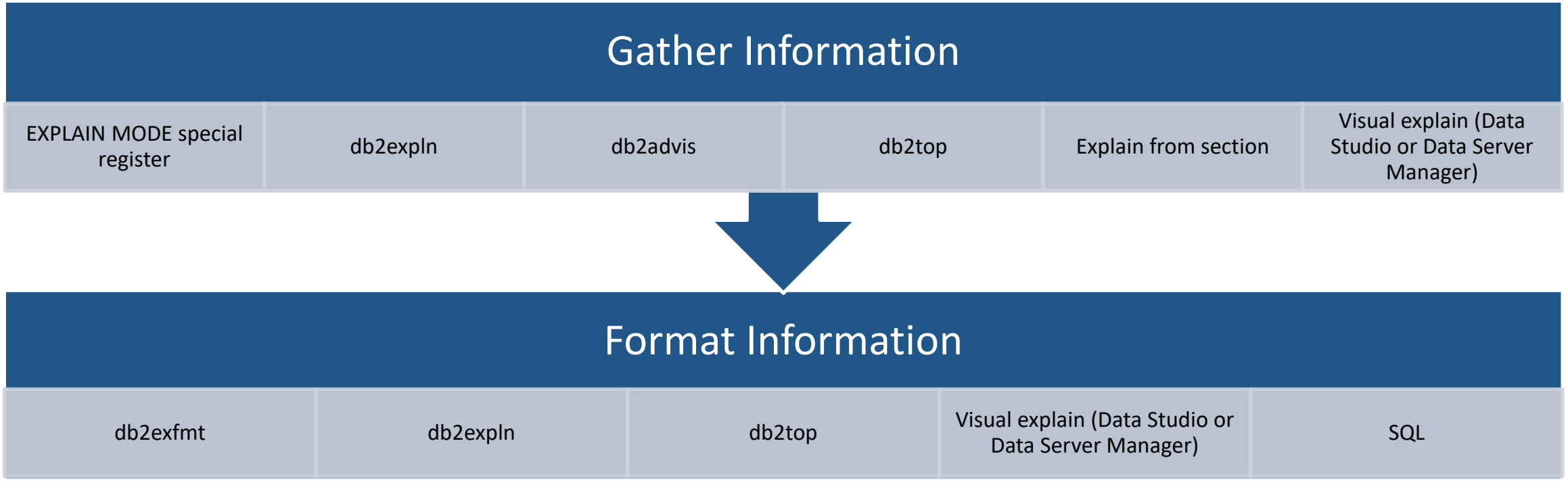
- Explain (verb): to generate a graphical or tabular layout of the access plan used to retrieve and return the data used to satisfy a query
- Example: Have you explained that problem query yet?

- Explain (adjective): having to do with the facility to produce the graphical or tabular layout of the access plan used to retrieve and return the data used to satisfy a query
- Examples: Explain Plan, Explain Information, Explain Tables

- Explain (noun): short for Explain Plan
- Example: Did you see the Explain on that query?

# Access Plan Information

- Access plans chosen by the optimizer can be optionally written out to explain tables
- Explain tables can be queried and formatted in various ways to understand how data is being accessed






XTIVIA

# Creating Explain Tables

- SYSINSTALLOBJECTS
  - Stored procedure which can create or delete explain tables – can specify schema and tablespace
  - SYSPROC schema

```
>>-SYSINSTALLOBJECTS--(--tool-name--,--action--,--tablespace-name--,--schema-name--)--><
```

  - Tool name is ‘EXPLAIN’
  - Possible actions:
    - C – Create
    - D – Drop
    - V – Verify
    - M – Migrate (use after upgrade)
  - If table space name is not specified, SYSTOOLSPACE is used for creation of explain tables
  - If schema name is not specified, SYSTOOLS is used
- Script also exists, but is not the preferred method
  - ~/sqllib/misc/EXPLAIN.DDL

# Using SYSINSTALLOBJECTS for Explain Tables

- Creating explain tables

```
db2inst1@ubuntu:~/sqllib/misc$ db2 "call SYSPROC.SYSINSTALLOBJECTS('EXPLAIN','C',NULL,NULL)"

  Return Status = 0
db2inst1@ubuntu:~/sqllib/misc$
```

- Verifying explain tables

```
db2inst1@ubuntu:~/sqllib/misc$ db2 "call SYSPROC.SYSINSTALLOBJECTS('EXPLAIN','V',NULL,NULL)"

  Return Status = 0
db2inst1@ubuntu:~/sqllib/misc$ _
```

- Dropping explain tables

```
db2inst1@ubuntu:~/sqllib/misc$ db2 "call SYSPROC.SYSINSTALLOBJECTS('EXPLAIN','D',NULL,NULL)"

  Return Status = 0
db2inst1@ubuntu:~/sqllib/misc$ _
```

# Errors Creating Explain Tables (1)

- If BLOCKNONLOGGED is set, you must use SYSINSTALLOBJECTS or disable BLOCKNONLOGGED temporally. Otherwise, you may see this error:

```
db2inst1@ubuntu:~/sqllib/misc$ db2 -tf EXPLAIN.DDL

******* IMPORTANT **********

USAGE: db2 -tf EXPLAIN.DDL

******* IMPORTANT **********


DB20000I  The UPDATE COMMAND OPTIONS command completed successfully.

DB20000I  The SQL command completed successfully.

DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL20054N  The operation was not performed because the table is in an invalid
state for the operation. Table name: "lob options prohibited with
blocknonlogged enabled". Reason code: "".  SQLSTATE=55019

DB21034E  The command was processed as an SQL statement because it was not a
```



XTIVIA

# Errors Creating Explain Tables (2)

- If the current EXPLAIN MODE is set to EXPLAIN:

```
db2inst1@ubuntu:~/sqllib/misc$ db2 "call SYSPROC.SYSINSTALLOBJECTS('EXPLAIN','C',NULL,NULL)"
SQL0219N  The required Explain table "DB2INST1.EXPLAIN_INSTANCE" does not
exist.  SQLSTATE=42704
db2inst1@ubuntu:~/sqllib/misc$ db2 set current explain mode no
DB20000I  The SQL command completed successfully.
db2inst1@ubuntu:~/sqllib/misc$ db2 "call SYSPROC.SYSINSTALLOBJECTS('EXPLAIN','C',NULL,NULL)"

  Return Status = 0
```

# Explain Methods

# Recommended Explain Methods for Gather and Format

## Gather

### EXPLAIN MODE

- Handles parameter markers well with explain mode of EXPLAIN
- Can execute SQL/DML at command line or from file
- Explain any dynamic statement
- No GUI needed

### EXPLAIN_FROM_SECTION

- Gets statement from package cache
- Works for static or dynamic statements that have been executed fairly recently
- Easy when working with MON_GET_PKG_CACHE_STMT

## Format

### db2exfmt

- Generates text-based graph
- Includes wealth of information in text-based format
- Hard to share with non-DBAs

# Recommended Explain Methods for Combined Gather and Format

## DSM Visual Explain

- Nice visual formatting, easy to export to sharable image
- Click for more information
- Features for formatting and searching explain plan
- May require remote connection
- GUI
- SQL may be entered or selected from monitor data
- Can only be saved for later as image

## db2top/dsmtop

- Same formatting as db2exfmt
- Can alternately use db2expln formatting
- SQL can be selected from monitoring data
- Easy to save output for later review/use

# Ember’s Favorite Explain Method

```
db2inst1@ubuntu:~$ db2 connect to sample

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.1.1.1
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2inst1@ubuntu:~$ db2 set current explain mode explain
DB20000I  The SQL command completed successfully.
db2inst1@ubuntu:~$ db2 "select sex, sum(sales) as total_sales from sales s join employee e on s.sales_person=e.lastna
me group by e.sex"
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604
db2inst1@ubuntu:~$ db2exfmt -d sample -1 -o sales_query.exfmt
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

Connecting to the Database.
Connect to Database Successful.
Using SYSTOOLS schema for Explain tables.
Output is in sales_query.exfmt.
Executing Connect Reset -- Connect Reset was Successful.
```

# db2exfmt Output Example

```
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991,
2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool



********************* EXPLAIN INSTANCE *********************

DB2_VERSION:        11.01.1
FORMATTED ON DB:    SAMPLE
SOURCE_NAME:        SQLC2O26
SOURCE_SCHEMA:      NULLID
SOURCE_VERSION:
EXPLAIN_TIME:       2017-01-06-13.52.49.175796
EXPLAIN_REQUESTER: DB2INST1

Database Context:
----------------
```

# Explain Modes

| NO | YES | EXPLAIN | EXPLAIN NORCAC | RECOMMEND INDEXES | EVALUATE INDEXES |
|---|---|---|---|---|---|
| • Collects no explain information<br>• Executes the DML<br>• Default value | • Collects explain information<br>• Executes the DML | • Collects explain information<br>• Does not execute the DML | • Collects explain information<br>• Does not execute the DML<br>• Explain information is generated as if Row and Column Access Control (RCAC) was not activated | • Collects explain information<br>• Does not execute the DML<br>• Recommends indexes by populating the ADVISE_INDEX table. | • Collects explain information<br>• Does not execute the DML<br>• Explain information generated can be manipulated by altering the USE_INDEX column of the ADVISE_INDEX table |



XTIVIA

# db2exfmt Syntax




XTIVIA

# Useful db2exfmt Options

- -d must be used to specify the database name
- -1 uses these options:
  - -e % ($USER is used for the explain table schema, or SYSTOOLS)
  - -n%
  - -v%
  - -w -1 (latest explain request)
  - -# 0 (all sections)
- -o can be used to specify an output file. Default output is to the terminal

# Error Running Explains

- Issue with binding:

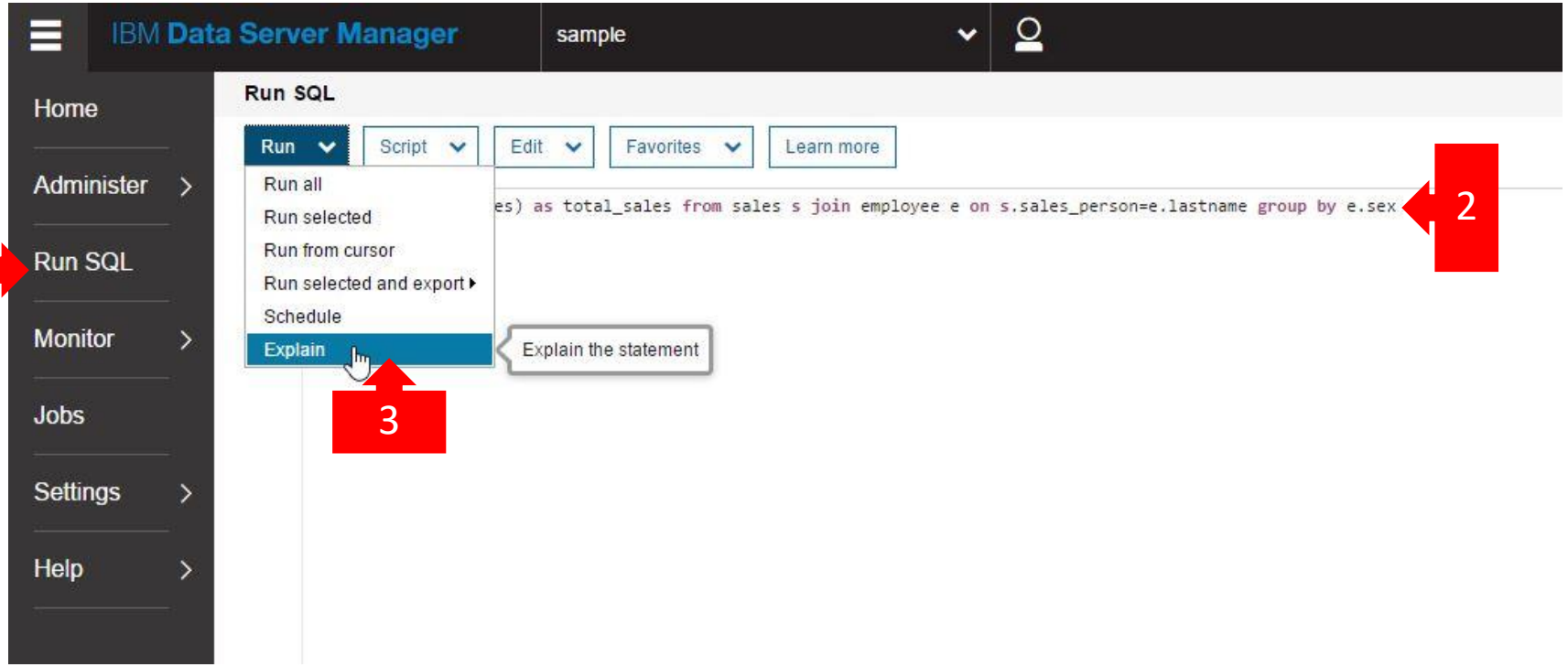
```
db2inst1@ubuntu:~/sqllib/misc$ db2exfmt -d sample -1 -o sales_query.exfmt
DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

Connecting to the Database.
Connect to Database Successful.
Binding package - Error during Bind, near line 7521.

Error Message =
SQL0035N  The file "db2exfmt.msg" cannot be opened.


SQLCA
Size     = 136
SQLCODE = -35
Tokens   = db2exfmt.msg
Function= sqlabndx
RC       = 0x0000 = 0
Reason   = 0x0000 = 0
Reason2 = 0x0000 = 0
Warning flags =

Error during bind.
Bind messages can be found in db2exfmt.msg
db2inst1@ubuntu:~/sqllib/misc$
```

- Resolution:

```
cd ~/sqllib/bnd
db2 bind db2exfmt.bnd blocking all grant public
```

# Visual Explain from DSM

1. Open DSM and select “Run SQL”
2. Type query or open query file
3. Click Run at the top, and select “Explain”

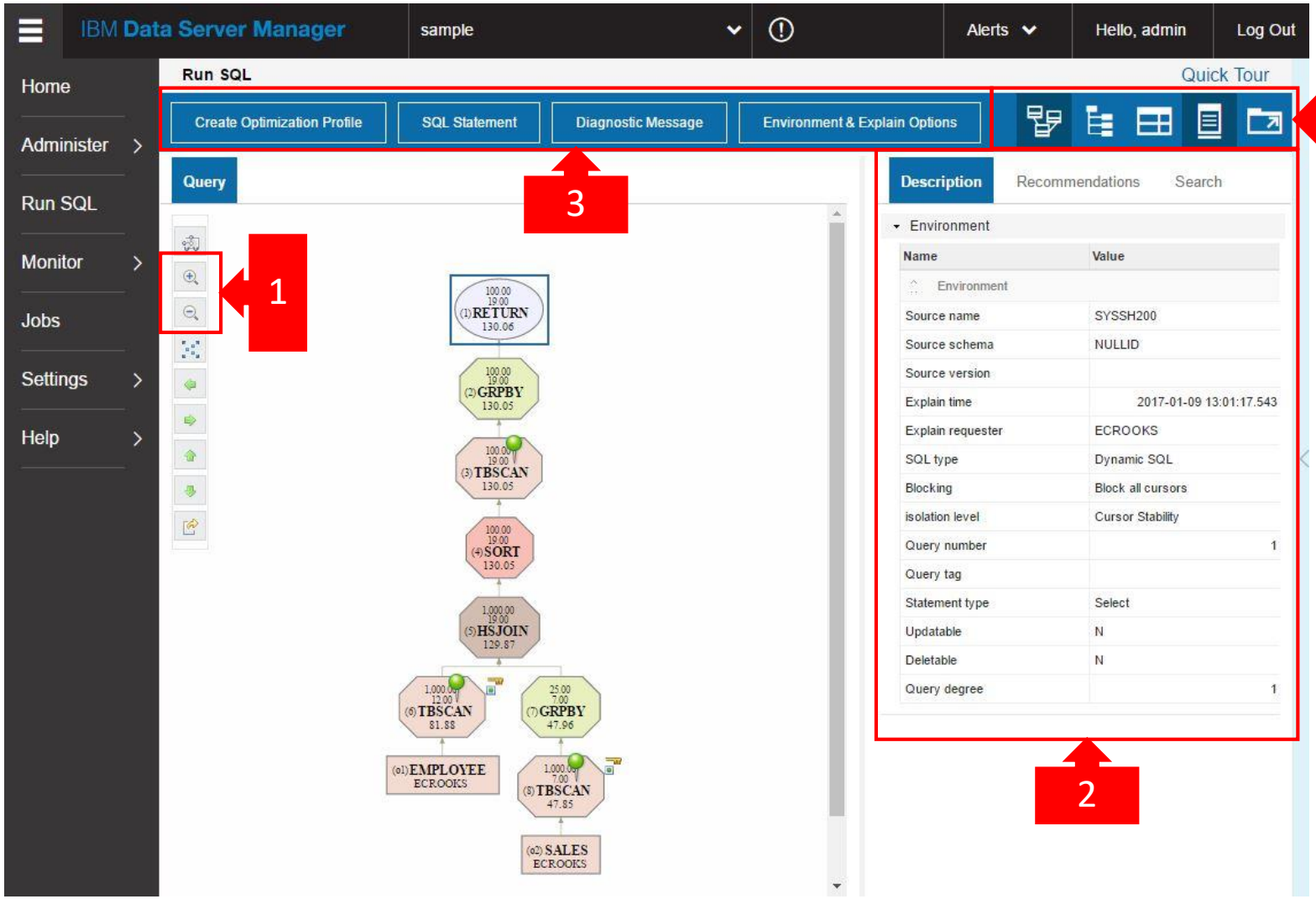





XTIVIA

# Visual Explain from DSM

1. You can zoom in and out to focus on smaller portions of the explain plan
2. A pane at the right provides tabs for:
   - Details as you click on each operator/object (Description)
   - Recommendations on areas for improvement (Recommendations)
   - Search feature that is useful for very large explains (Search)
3. Options at the top to
   - Create Optimization Profile
   - Show the SQL Statement
   - Show any messages for this explain
   - Show environment/explain options
4. Options on upper right to show in a tree or tabular view that may be more similar to other RDMS's explain output



# EXPLAIN MODE & Query Explain Tables

- Starts the same as method 1:

```
db2inst1@ubuntu:~$ db2 connect to sample

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.1.1.1
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2inst1@ubuntu:~$ db2 set current explain mode explain
DB20000I  The SQL command completed successfully.
db2inst1@ubuntu:~$ db2 "select sex, sum(sales) as total_sales from sales s join employee e on s.sales_person=e.lastna
me group by e.sex"
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604
```

- Format output by creating view and running query
- View syntax available in the student notes, or here: https://github.com/fatalmind/DB2-last-explained/

# Query Explain Tables

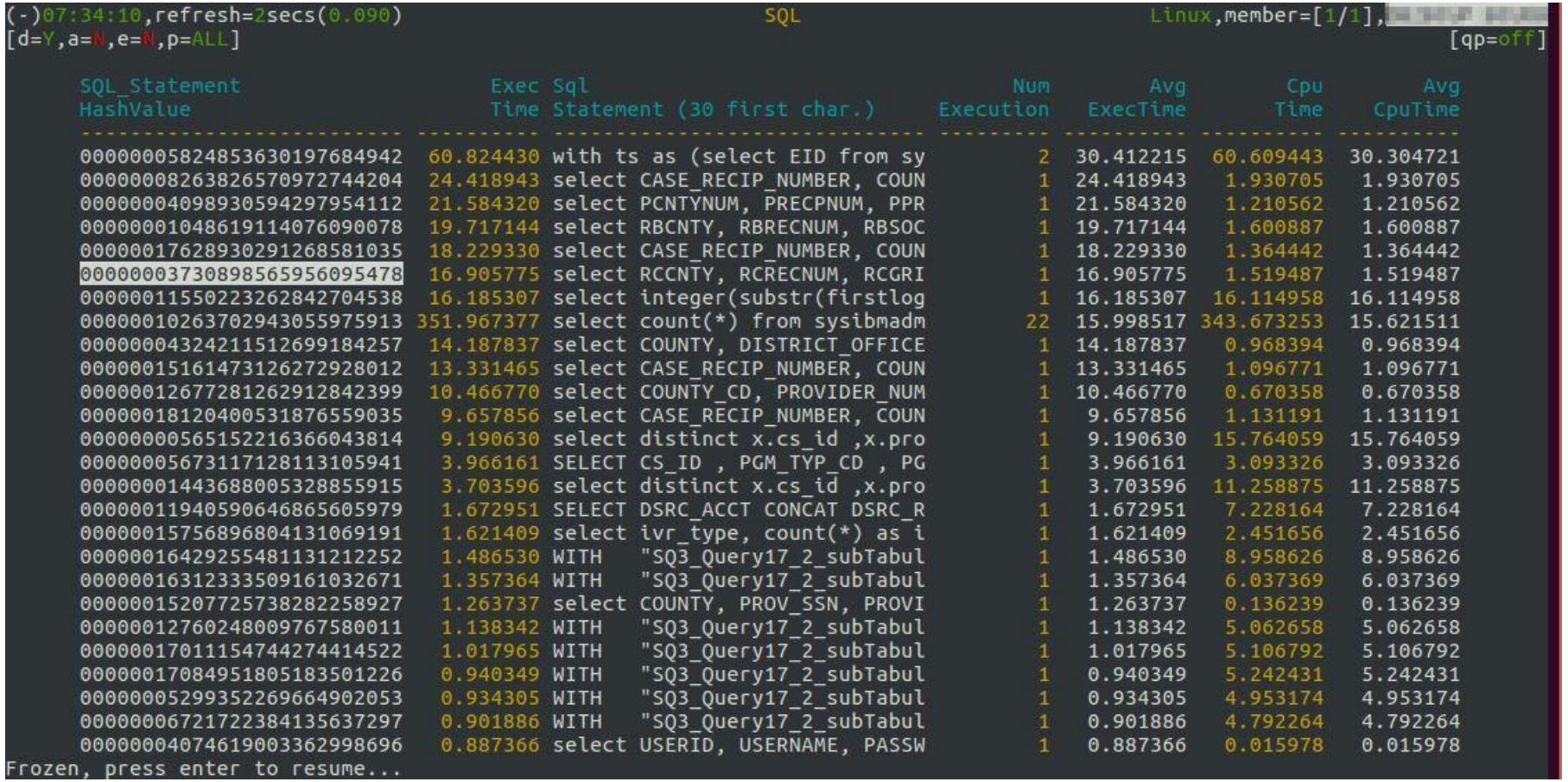
- Markus Winand of http://use-the-index-luke.com/

```
db2inst1@ubuntu:~$ db2 "select * from last_explained"

Explain Plan
-------------------------------------------------------------------------------
ID | Operation             |                 Rows | Cost
 1 | RETURN                |                      |   13
 2 |  GRPBY (FINAL)        |    2 of 2 (100.00%) |   13
 3 |   TBSCAN              |    2 of 2 (100.00%) |   13
 4 |    SORT (PARTIAL)     |  2 of 42 (  4.76%) |   13
 5 |     HSJOIN            |            42 of 41 |   13
 6 |      TBSCAN EMPLOYEE | 42 of 42 (100.00%) |    6
 7 |      TBSCAN SALES    | 41 of 41 (100.00%) |    6

Predicate Information
 5 - JOIN (Q2.SALES_PERSON = Q1.LASTNAME)

Explain plan (c) 2014-2016 by Markus Winand - NO WARRANTY - V20161027
Modifications by Ember Crooks - NO WARRANTY
http://use-the-index-luke.com/s/last_explained

  15 record(s) selected.
```

# Explain from db2top (1)

```
db2top -d sample
```

- Shift-D for dynamic SQL
- Choose a query and copy the SQL_Statement HashValue

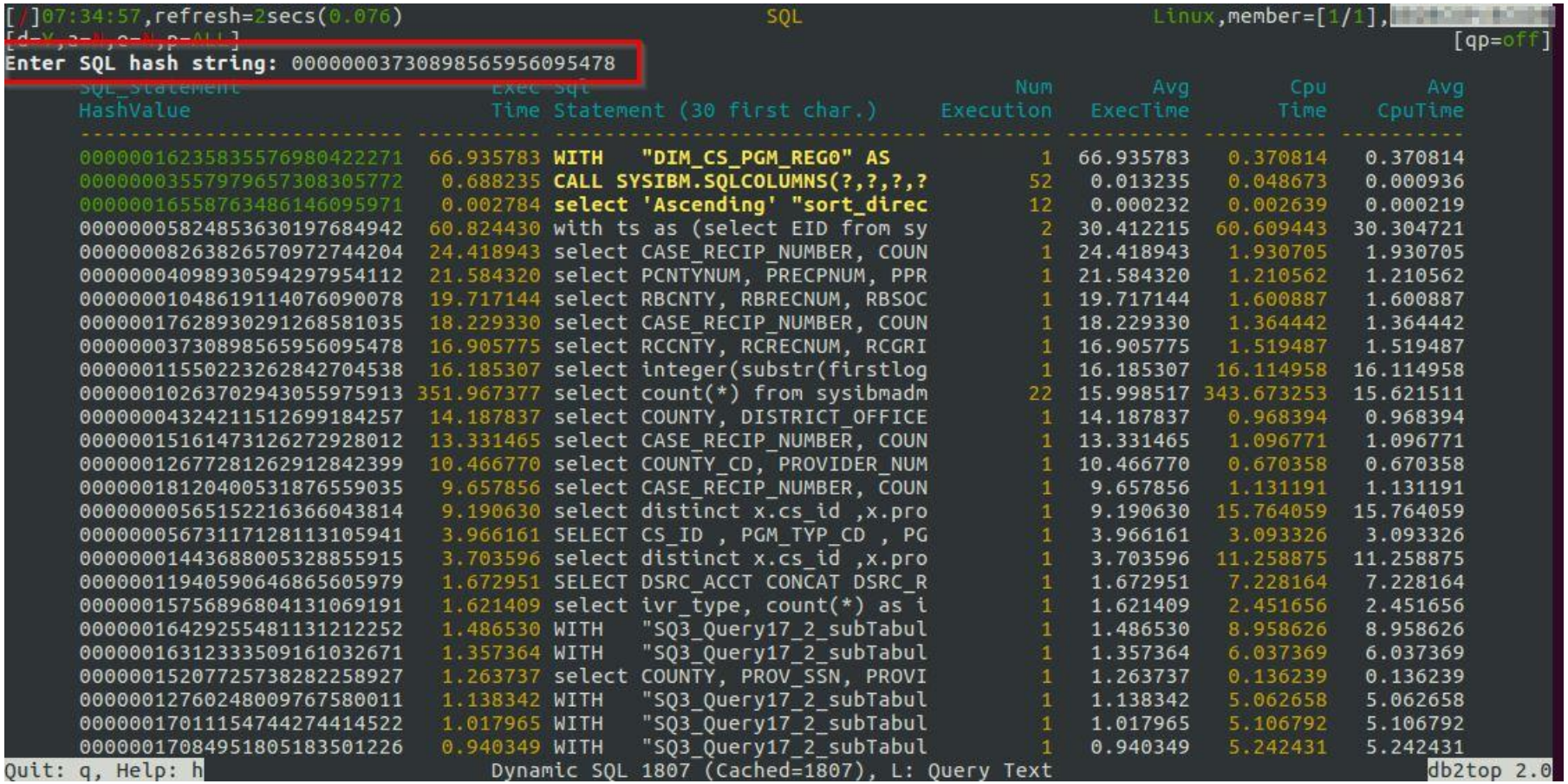





XTIVIA

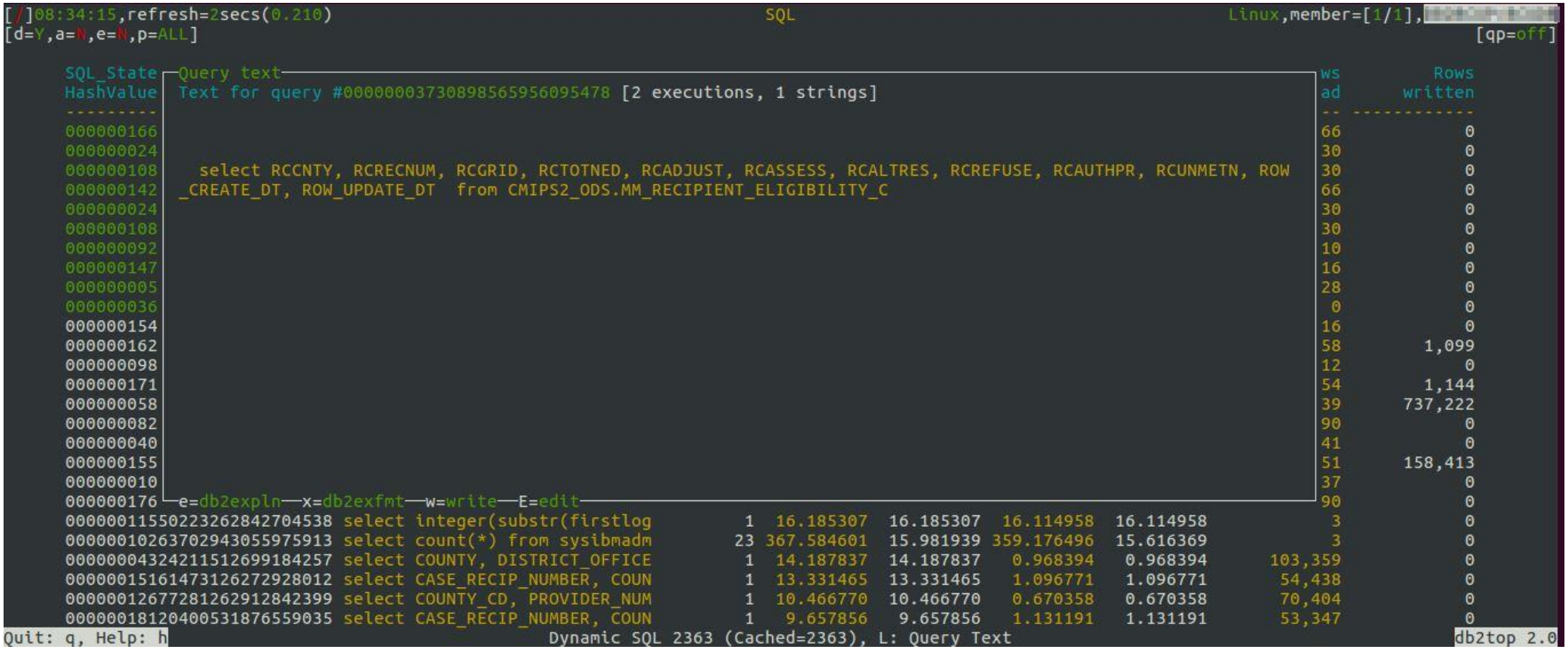
# Explain from db2top (2)

- Shift-L for statement text
- Enter the SQL_Statement HashValue



# Explain from db2top (3)

- x to generate explain output in db2exfmt format



# Explain from db2top (4)

- Explain displayed is opened in vi
- :w filename.exfmt
- :q! to get back to db2top

```
DB2 Universal Database Version 10.5, 5622-044 (c) Copyright IBM Corp. 1991, 2012
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool



****************** EXPLAIN INSTANCE ******************

DB2_VERSION:       10.05.7
FORMATTED ON DB:   BCUDB
SOURCE_NAME:       SQLC2K26
SOURCE_SCHEMA:     NULLID
SOURCE_VERSION:
EXPLAIN_TIME:      2017-01-11-08.37.51.010776
EXPLAIN_REQUESTER: VDBA

Database Context:
----------------
        Parallelism:          Intra-Partition Parallelism
        CPU Speed:            2.401083e-07
        Comm Speed:           0
        Buffer Pool size:     382480
        Sort Heap size:       500000
        Database Heap size:   260000
        Lock List size:       100000
        Maximum Lock List:    98
        Average Applications: 1
        Locks Available:      3136000

Package Context:
---------------
        SQL Type:           Dynamic
"/tmp/explain.5154" 342L, 8219C
```

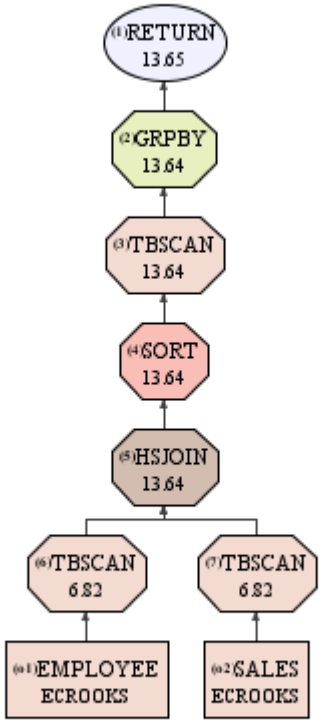
# Reading Explain Plans

# Explain Questions

How do I make this query run faster?

- Which operators are most expensive?
- Which branches of the explain plan are most expensive?
- Are there any problematic operators?
- Is DB2 accessing the data in a logical way?
- Is DB2 using an index you expect to be used?
- Are the row count estimates about right?
- On a grand scale, is the query cheap or expensive?

# Visual Explain Plan vs. db2exfmt Explain Plan




```
                Rows
               RETURN
               (   1)
                Cost
                 I/O
                  |
                 7.4
               GRPBY
               (   2)
               13.7118
                  2
                  |
                 7.4
               TBSCAN
               (   3)
               13.7115
                  2
                  |
                 7.4
               SORT
               (   4)
               13.7111
                  2
                  |
                  74
               HSJOIN
               (   5)
               13.7021
                  2
              /---+----\
            74           25
          TBSCAN       GRPBY
          (   6)       (   7)
          6.82622      6.87242
            1            1
            |            |
            74           145
      TABLE: DB2INST1  TBSCAN
         EMPLOYEE      (   8)
            Q1         6.84066
                          1
                          |
                          145
                    TABLE: DB2INST1
                       SALES
                        Q2
```

# Sections of db2exfmt Output

- Context and Settings
  - Database Context
  - Package Context
- Statement
  - Original Statement
  - Optimized Statement
- Access Plan (graph)
- Operator Symbols
- Extended Diagnostic Information
- Plan Details
- Objects Used in Access Plan

# db2exfmt – Context and Settings

- Information provided includes:
  - DB2 version
  - Time of explain
  - Whether intra-partition parallelism is enabled
  - Settings for critical memory areas and configuration parameters
  - Optimization level
  - Isolation Level

```
DB2_VERSION:        10.05.7
FORMATTED ON DB:    SAMPLE
SOURCE_NAME:        SQLC2K26
SOURCE_SCHEMA:      NULLID
SOURCE_VERSION:
EXPLAIN_TIME:       2017-01-11-07.48.28.307908
EXPLAIN_REQUESTER: ECROOKS

Database Context:
----------------

 Parallelism:          Intra-Partition Parallelism
        CPU Speed:             2.401083e-07
        Comm Speed:            0
        Buffer Pool size:      382480
        Sort Heap size:        500000
        Database Heap size:    260000
        Lock List size:        100000
        Maximum Lock List:     98
        Average Applications: 1
        Locks Available:       3136000

Package Context:
---------------
        SQL Type:           Dynamic
        Optimization Level: 5
        Blocking:           Block All Cursors
        Isolation Level:    Cursor Stability



---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:       1
        QUERYTAG:      CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     -1
```

# db2exfmt - Statement

- Original statement is included
- Optimized statement is an internal representation of how DB2 rewrote the query and may not be directly executable

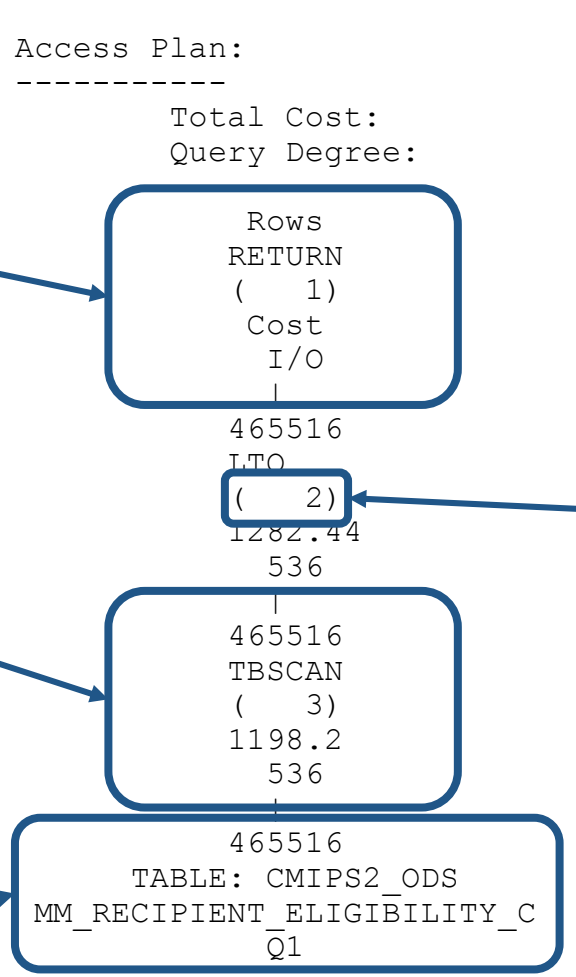
```
Original Statement:
------------------
select
  RCCNTY,
  RCRECNUM,
  RCGRID,
  RCTOTNED,
  RCADJUST,
  RCASSESS,
  RCALTRES,
  RCREFUSE,
  RCAUTHPR,
  RCUNMETN,
  ROW_CREATE_DT,
  ROW_UPDATE_DT
from
  CMIPS2_ODS.MM_RECIPIENT_ELIGIBILITY_C

Optimized Statement:
-------------------
SELECT
  Q1.RCCNTY AS "RCCNTY",
  Q1.RCRECNUM AS "RCRECNUM",
  Q1.RCGRID AS "RCGRID",
  Q1.RCTOTNED AS "RCTOTNED",
  Q1.RCADJUST AS "RCADJUST",
  Q1.RCASSESS AS "RCASSESS",
  Q1.RCALTRES AS "RCALTRES",
  Q1.RCREFUSE AS "RCREFUSE",
  Q1.RCAUTHPR AS "RCAUTHPR",
  Q1.RCUNMETN AS "RCUNMETN",
  Q1.ROW_CREATE_DT AS "ROW_CREATE_DT",
  Q1.ROW_UPDATE_DT AS "ROW_UPDATE_DT"
FROM
  CMIPS2_ODS.MM_RECIPIENT_ELIGIBILITY_C AS Q1
```

# Basic Layout of Graph

The first item in the access plan shows which positions represent:
- number of rows
- name of the operator (RETURN)
- cost in timerons
- number of I/Os

This operator is a Table Scan (TBSCAN)
It expects 465,516 rows
Cumulatively, to this point in the graph
- 1198.2 Timerons have been used
- 536 I/Os have been done

This represents an object, in this case a table.
Information provided includes:
- Schema
- Table Name
- Cardinality




Total cost of query, in timerons

Number of this operator, which can be used to correlate details in the “Plan Details” section



XTIVIA

# Plan Details (1)

- Each operator in the explain plan has a lot of details in the plan details section
- Cumulative costs to this point in the access plan
- Arguments - list of assumptions made by the optimizer

```
3) TBSCAN: (Table Scan)
        Cumulative Total Cost:          1198.2
        Cumulative CPU Cost:            8.7997e+08
        Cumulative I/O Cost:            536
        Cumulative Re-Total Cost:       192.476
        Cumulative Re-CPU Cost:         8.0162e+08
        Cumulative Re-I/O Cost:         0
        Cumulative First Row Cost:      6.89851
        Estimated Bufferpool Buffers:   536

        Arguments:
        ---------
        CUR_COMM: (Currently Committed)
                TRUE
        LCKAVOID: (Lock Avoidance)
                TRUE
        MAXPAGES: (Maximum pages for prefetch)
                ALL
        PREFETCH: (Type of Prefetch)
                SEQUENTIAL
        ROWLOCK : (Row Lock intent)
                SHARE (CS/RS)
        SCANDIR : (Scan Direction)
                FORWARD
        SCANGRAN: (Intra-Partition Parallelism Scan Granularity)
                2
        SCANTYPE: (Intra-Partition Parallelism Scan Type)
                LOCAL PARALLEL
        SCANUNIT: (Intra-Partition Parallelism Scan Unit)
                PAGE
        SKIP_INS: (Skip Inserted Rows)
                TRUE
        SPEED   : (Assumed speed of scan, in sharing structures)
                FAST
        TABLOCK : (Table Lock intent)
                INTENT SHARE
        TBISOLVL: (Table access Isolation Level)
                CURSOR STABILITY
        THROTTLE: (Scan may be throttled, for scan sharing)
                TRUE
        VISIBLE : (May be included in scan sharing structures)
                TRUE
        WRAPPING: (Scan may start anywhere and wrap)
                TRUE
```



XTIVIA

# Plan Details (2)

- Input stream defines columns and rows coming into this operator
- Output stream defines columns and rows going out of this operator

```
Input Streams:
-------------
        1) From Object CMIPS2_ODS.MM_RECIPIENT_ELIGIBILITY_C

                Estimated number of rows:       465516
                Number of columns:              13
                Subquery predicate ID:          Not Applicable

                Column Names:
                ------------
                +Q1.$RID$+Q1.ROW_UPDATE_DT+Q1.ROW_CREATE_DT
                +Q1.RCUNMETN+Q1.RCAUTHPR+Q1.RCREFUSE
                +Q1.RCALTRES+Q1.RCASSESS+Q1.RCADJUST
                +Q1.RCTOTNED+Q1.RCGRID+Q1.RCRECNUM+Q1.RCCNTY

Output Streams:
--------------
        2) To Operator #2

                Estimated number of rows:       465516
                Number of columns:              12
                Subquery predicate ID:          Not Applicable

                Column Names:
                ------------
                +Q2.ROW_UPDATE_DT+Q2.ROW_CREATE_DT+Q2.RCUNMETN
                +Q2.RCAUTHPR+Q2.RCREFUSE+Q2.RCALTRES
                +Q2.RCASSESS+Q2.RCADJUST+Q2.RCTOTNED+Q2.RCGRID
                +Q2.RCRECNUM+Q2.RCCNTY
```

# Objects Used in Access Plan

- Includes details about the object (index, table, etc) being accessed
- Heavily dependent on current runstats

```
Objects Used in Access Plan:
---------------------------

        Schema: CMIPS2_ODS
        Name:   MM_RECIPIENT_ELIGIBILITY_C
        Type:   Table
                    Time of creation:              2015-01-09-14.56.53.714157
                    Last statistics update:        2017-01-06-14.59.19.358097
                    Number of columns:              12
                    Number of rows:                 465516
                    Width of rows:                  69
                    Number of buffer pool pages:    536
                    Number of data partitions:      1
                    Distinct row values:            No
                    Tablespace name:                CMIPS_TS16K
                    Tablespace overhead:            6.725000
                    Tablespace transfer rate:       0.160000
                    Source for statistics:          Single Node
                    Prefetch page count:            4
                    Container extent page count:    4
                    Table overflow record count:    0
                    Table Active Blocks:            -1
                    Average Row Compression Ratio:  4.6663
                    Percentage Rows Compressed:     100
                    Average Compressed Row Size:    15
```

# Common Explain Operators

| RETURN | TBSCAN | IXSCAN | FETCH |
| --- | --- | --- | --- |
| Return of data from the query | Table scan | Index Scan | Fetch rows/columns from a table |
| **HSJOIN** | **NLJOIN** | **SORT** | **GRPBY** |
| Hash join | Nested loop join | Sorting of rows | Grouping of rows |
| **FILTER** | **LTQ** | **CTQ** | **GENROW** |
| Filtering of rows | Table queue for intra-partition parallelism | Table queue for column-organized data | Generation of data, likely an in-list or insert |

# Timerons

- Unit used to measure overall cost of an access plan and each step in the plan
- Used by the optimizer to compare the cost of various possible access plans to select the most efficient plan
- Abstract unit of measure
- Cannot be converted to actual units of time
- Actual time depends on the hardware on which the query is executed

# Commercial Break



XTIVIA

# Collecting Section Actuals

# Selectivity and Filter Factors

- The DB2 Optimizer estimates how many rows it expects at each step
- Row Estimates influence:
  - Query cost
  - Join types chosen
  - Access plan chosen
  - Memory allocation
- Filter Factor reported in the explain plan
- Filter Factor * number of rows input from last operator = expected number of rows
- Filter Factor is calculated:
  - Using runstats
  - Assuming a normal distribution of data if runstats are not available
  - Assuming a normal distribution of data if parameter markers or host variables are used

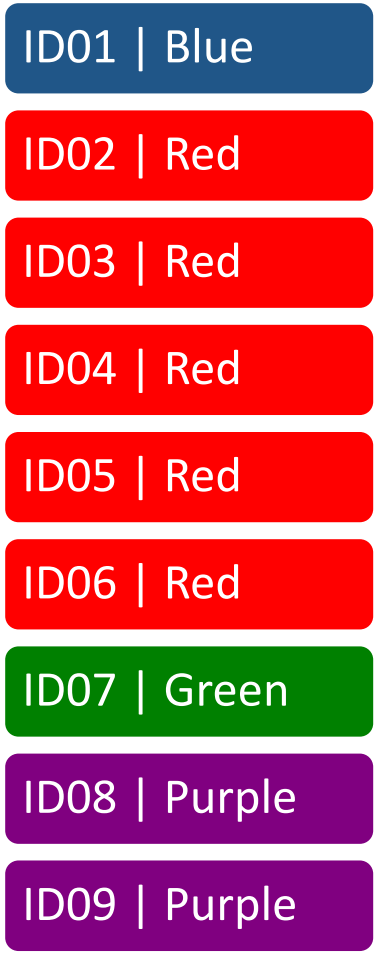
# Filter Factors

```
Predicates:
----------
2) Sargable Predicate,
        Comparison Operator:            Equal (=)
        Subquery Input Required:        No
        Filter Factor:                  0.600499

        Predicate Text:
        --------------
        (Q1.FIELD1 = 'ET_SUBMIT')

3) Sargable Predicate,
        Comparison Operator:            Not Applicable
        Subquery Input Required:        No
        Filter Factor:                  0.00143409

        Predicate Text:
        --------------
        ((Q1.FIELD2 = 'QUEUED') OR
         ((Q1.FIELD2 = 'FAILED') AND
          (Q1.FIELD4 <= +2.00000000000000E+000)))
```

```
Input Streams:
-------------
        1) From Object DB2INST1.XORDERDETAIL
                Estimated number of rows:       2.21443e+06
                Number of columns:              6
                Subquery predicate ID:          Not Applicable

                Column Names:
                ------------
                +Q1.$RID$+Q1.FIELD7+Q1.ORDERS_ID+Q1.FIELD4
                +Q1.FIELD2+Q1.FIELD1
Output Streams:
--------------
        2) To Operator #3

                Estimated number of rows:       1907
                Number of columns:              2
                Subquery predicate ID:          Not Applicable

                Column Names:
                ------------
                +Q1.FIELD7+Q1.ORDERS_ID
```

0.600499 * 0.00143409 * 2214430 = 1907

# Section Actuals




- Sometimes a performance problem is caused by a difference between estimated data returned and actual data returned.

- `select * from table where color='Red'`
  - Filter Factor = 0.25
  - Estimated rows = 2.25
  - Actual rows = 5
- `select * from table where color='Blue'`
  - Filter Factor = 0.25
  - Estimated rows = 2.25
  - Actual rows = 1



XTIVIA

# Factors in Estimation

- Runstats
  - Current?
  - Detailed statistics
  - Column group statistics
  - Statistical views
- Use of parameter markers
  - Saves time on compilation, but does it save time if the data is skewed?

# db2caem - Syntax

- Tool that collects not just explain plan, but also the actuals at each step
- Basic syntax:
  - `db2caem -d <db_name> -o <output_dir> -tbspname <admin_ts> -sf <query_file>`
- Example:
  - `mkdir query1_caem`
  - `db2caem -d SAMPLE -o query1_caem –tbspname dba32k -sf query1.sql`

- Full syntax diagram: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.admin.cmd.doc/doc/r0057282.html

# db2caem Actions

1. Creates an independent set of explain tables in the SYSTOOLS tablespace (cannot be changed)
2. Creates the event monitor for activities and the 4 associated tables in the tablespace
3. Uses the stored procedure WLM_SET_CONN_ENV to gather information to the activity event monitor on anything done from this connection
4. Runs the statement that you gave it (no parameter markers)
5. Uses the stored procedure WLM_SET_CONN_ENV to stop gathering activity for the connection
6. Turns off the event monitor
7. Rolls back the statement that you gave it
8. Uses EXPLAIN_FROM_ACTIVITY stored procedure to capture explain information
9. Runs dozens and dozens of SQL statements on the explain tables, the statement event monitor tables, and the system catalog tables
10. Writes the explain plan with actuals to the output directory
11. Exports data from the event monitor tables to a subdirectory of the output directory
12. Drops the event monitor, the event monitor tables, and the explain tables

# db2caem Output

```
Access Plan:
-----------
        Total Cost:             570.128
        Query Degree:           1


                Rows
            Rows Actual
              RETURN
              (   1)
               Cost
                I/O
                |
              1.0123
                 0
              NLJOIN
              (   2)
              570.128
                NA
           /-------+-------\
       1.01244             0.999855
          0                    0
       TBSCAN               FETCH
       (   3)               (   4)
       554.792              15.1474
         NA                   NA
         |                 /---+----\
        6918          0.999855       6917
         NA               0           NA
   TABLE: WSCOMUSR     IXSCAN    TABLE: WSCOMUSR
      CATENTRY         (   5)       CATENTDESC
         Q1            7.5741           Q2
                         NA
                         |
                        6917
                         NA
                   INDEX: WSCOMUSR
                      I0001248
                         Q2
```



XTIVIA

# Section Actuals Without db2caem (1)

- Enable section actuals
  - db2 "CALL WLM_SET_CONN_ENV( NULL, '<collectactdata>WITH DETAILS, SECTION</collectactdata> <collectsectionactuals>BASE</collectsectionactuals>')“
- Capture section actuals
  - db2 "ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA on coordinator WITH DETAILS,SECTION“
- Create and activate activity event monitor
  - db2 "CREATE EVENT MONITOR ACTEVMON FOR ACTIVITIES WRITE TO TABLE“
  - db2 "set event monitor actevmon state=1“
- Generate explain plan
  - db2 set current explain mode yes
  - db2 -tvf query1_vals.sql

# Section Actuals Without db2caem (2)

```
$ db2 "select * from ecrooks.last_explained with ur"

Explain Plan
-----------------------------------------------------------------------------------------------
ID | Operation                      |                        Rows |  Cost
 1 | RETURN                         |                             | 35435
 2 |  TBSCAN                        |     2393 of 2393 (100.00%) | 35435
 3 |   SORT (UNIQUE)                |     2393 of 2393 (100.00%) | 35435
 4 |    HSJOIN (RIGHT)              |                2393 of 271 | 35434
 5 |     HSJOIN                     |      271 of 9645 (  2.81%) | 16764
 6 |      HSJOIN                    |   18061 of 42973 ( 42.03%) | 16261
 7 |       NLJOIN                   |                85967 of 6 |  8645
 8 |        FETCH ATTR              |           6 of 6 (100.00%) |     9
 9 |         IXSCAN I0001203        |          6 of 88 (  6.82%) |     0
10 |        FETCH CATENTRYATTR      |   14328 of 14328 (100.00%) |  1439
11 |         IXSCAN I0001463        | 14328 of 1260855 (  1.14%) |    91
12 |       FETCH CATENTRY           |   42973 of 42973 (100.00%) |  7612
13 |        RIDSCN                  |   42973 of 42973 (100.00%) |  2055
14 |         SORT (UNIQUE)          |   42973 of 42973 (100.00%) |  2055
15 |          IXSCAN I0000064       |  42973 of 204538 ( 21.01%) |  2045
16 |      TBSCAN ATTRVALDESC        |     9645 of 9645 (100.00%) |   501
17 |     HSJOIN                     |     2393 of 2393 (100.00%) | 18669
18 |      TBSCAN ATTRVAL            |     7489 of 7489 (100.00%) |   121
19 |      HSJOIN                    |     2393 of 2399 ( 99.75%) | 18548
20 |       FETCH ATTRVALDESC        |     7469 of 7469 (100.00%) |   531
21 |        IXSCAN I0001468         |     7469 of 9645 ( 77.44%) |    94
22 |       NLJOIN                   |                2399 of 1 | 18016
23 |        TBSCAN                  |     4032 of 4032 (100.00%) |  9688
24 |         SORT                   |     4032 of 4032 (100.00%) |  9688
25 |          HSJOIN                |    4032 of 20515 ( 19.65%) |  9687
26 |           NLJOIN               |               20515 of 1 |  2068
27 |            FETCH ATTR          |           1 of 1 (100.00%) |     7
28 |             RIDSCN             |           1 of 1 (100.00%) |     0
29 |              SORT (UNIQUE)     |           1 of 1 (100.00%) |     0
30 |               IXAND            |           1 of 6 ( 16.67%) |     0
31 |                IXSCAN I0001203 |          6 of 88 (  6.82%) |     0
…
```



XTIVIA

# Using the Design Advisor

# The DB2 Design Advisor

- Recommends indexes, MQTs, MDC Dimensions, and/or database partitions based on a supplied workload of queries/DML
- Separate tool from explain, but design advisor tables are created along with the explain tables
- Simulates indexes and calculates the cost of the workload as if the indexes existed
- Creates temporary simulation catalog tables while running

# The DB2 Design (Index) Advisor

## db2advis

- Easy to execute at the command line
- Free to use
- Easy to share with others who understand the index/design advisor output

## DSM “Tune Query”

- Cannot be used without creating a repository database
- Information presented in slightly different ways
- Past results automatically saved

# Index/Design Advisor

- Treats each query or set of queries you feed it as if they are the only workload against the database
- Sometimes you can come up with better solutions by analyzing the explain plan
- Favors index-only access, which is only occasionally appropriate
- May lead to over-indexing
- Recommends dropping indexes – don't take these suggestions unless you are really feeding it a complete workload with appropriate frequency values
- If many indexes are recommended, often a small subset will give you 95% of the benefit – important to analyze which indexes actually provide benefit
- Use common sense when adding indexes – avoid
  - Duplication of data - Indexes that are full duplicates of other indexes with different column orders, unless you are also dropping the other index
  - Indexes that are a subset of other indexes in the same order
  - Very wide indexes or indexes that contain every column of the table
  - Low-cardinality indexes

## Do

- Perform runstats before running the advisor
- Analyze indexing recommendations carefully
- Consider the relative importance of the performance of this query or workload
- Test your solutions thoroughly
- Consider dropping unused indexes after through analysis. Other useful tools for this include:
  - LASTUSED column of SYSCAT.INDEXES
  - Usage lists

## Don't

- Add indexes just because the advisor recommends them
- Drop indexes just because the advisor recommends them
- Analyze your whole workload and add and drop all indexes as recommended

# Running db2advis

- User must have the authority to execute the query being examined, and have read and write access on explain tables

```
db2inst1@ubuntu:~$ db2advis -d sample -i sales_query.sql |tee sales_query.advis

Using user id as default schema name. Use -n option to specify schema
execution started at timestamp 2017-01-16-09.35.49.142368
found [1] SQL statements from the input file
Recommending indexes...
```

sales_query.advis

# db2advis output

```
1  indexes in current solution
 [3277.0000] timerons  (without recommendations)
 [1680.0000] timerons  (with current solution)
 [48.73%] improvement
--
--
-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1],    0.915MB
   CREATE INDEX "DB2INST1"."IDX1701161736000" ON "DB2INST1"."SALES"
   ("SALES_PERSON" ASC, "SALES" DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
   COMMIT WORK ;
--
--
-- RECOMMENDED EXISTING INDEXES
-- ============================
--
--
-- UNUSED EXISTING INDEXES
-- -----------------------
-- DROP INDEX "DB2INST1"."XEMP2";
-- -----------------------------
--

-- ====ADVISOR DETAILED XML OUTPUT============
```

Cost in timerons before and after recommended solution
Percentage improvement that this represents

Space each index will take on disk

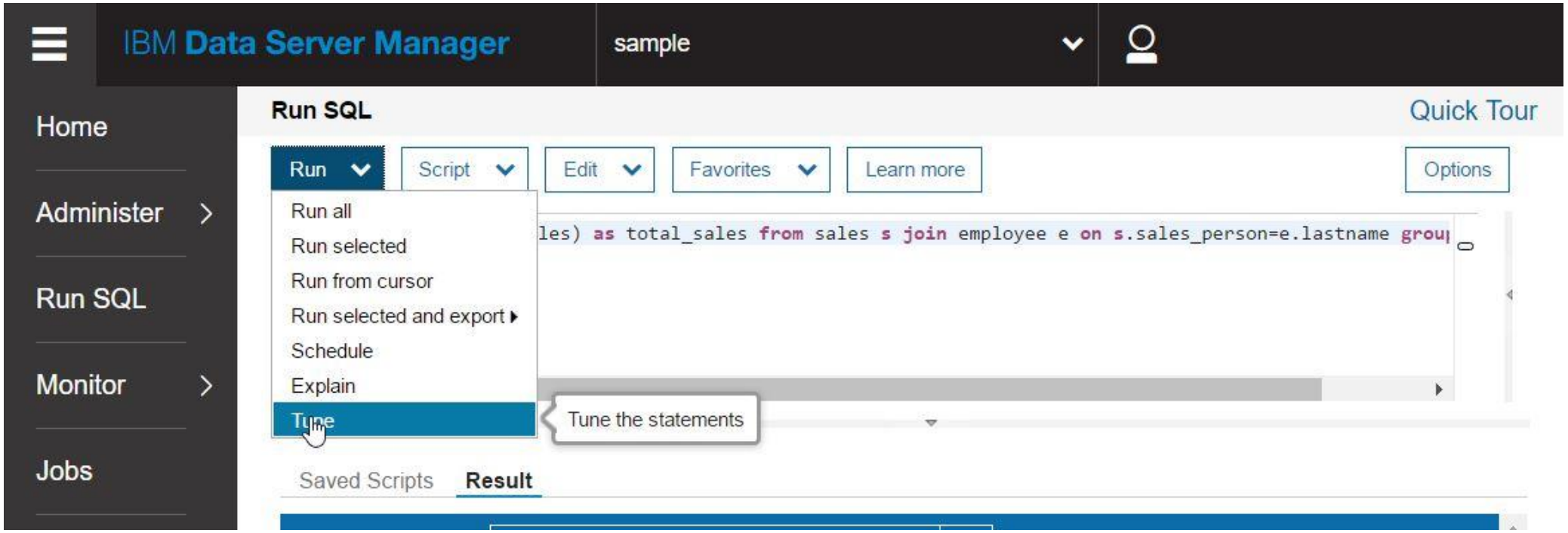
Complete syntax for creating each recommended index

- Alter index name to match your naming standard

Drop statement for each index not used by this workload, regardless of other usage

# Tuning Queries with DSM Query Tuning (1)

1. Open DSM and select “Run SQL”
2. Type query or open query file
3. Click Run at the top, and select “Tune”

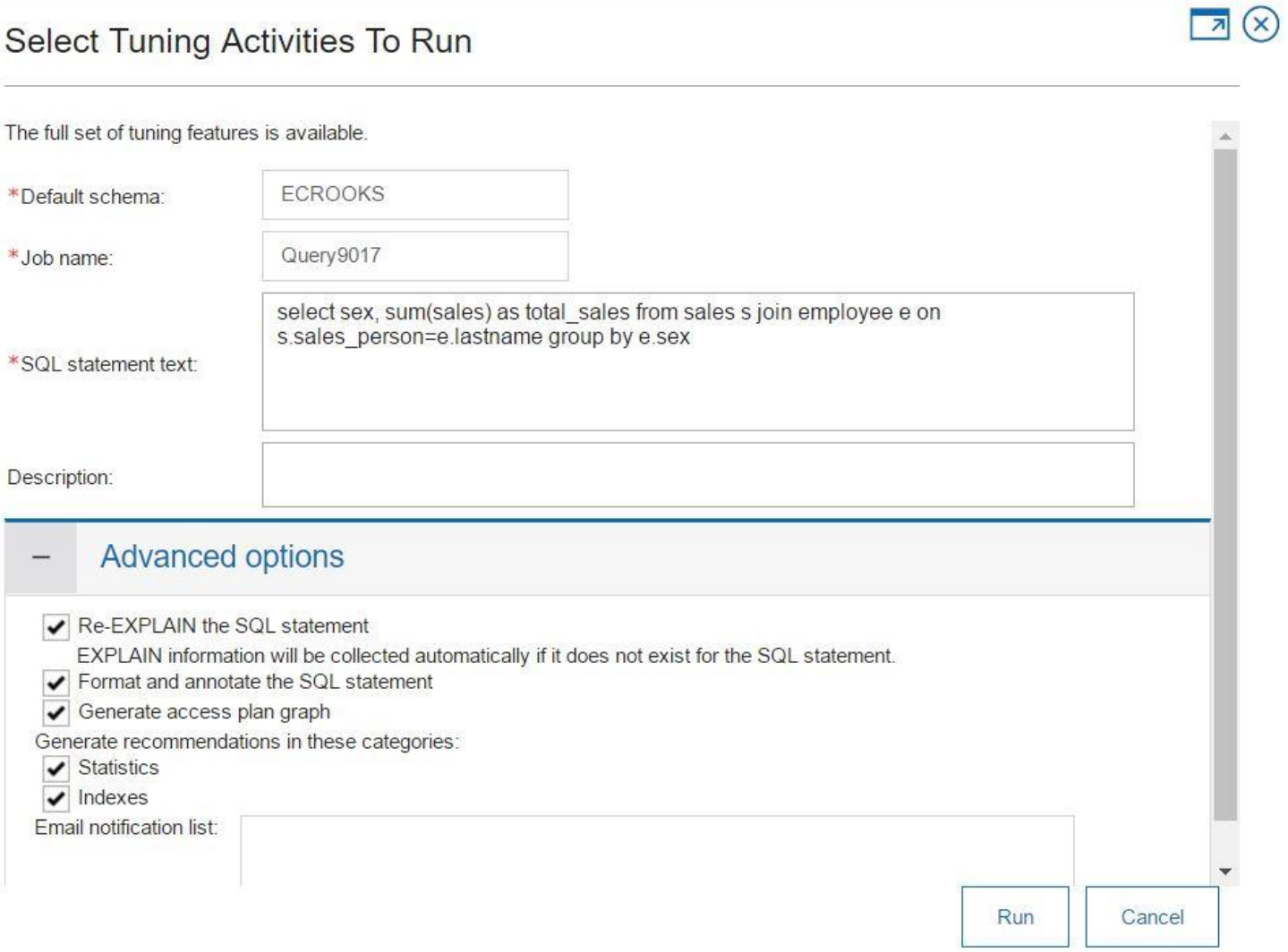





XTIVIA

# Tuning Queries with DSM Query Tuning (2)

4. Accept the defaults or make changes
5. Click “Run”

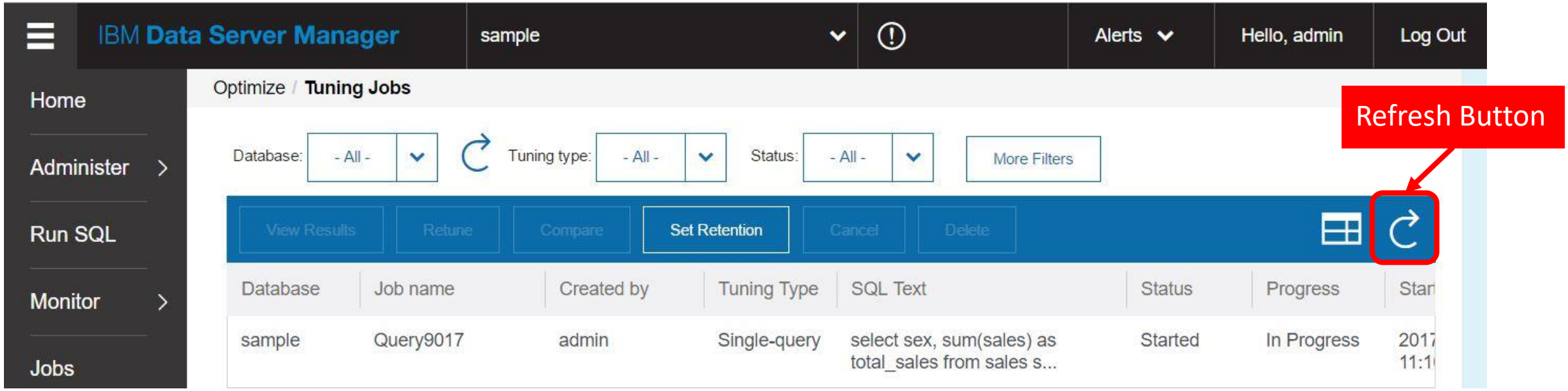





XTIVIA

# Tuning Queries with DSM Query Tuning (3)

6. Click the refresh button until the progress shows as “Completed”

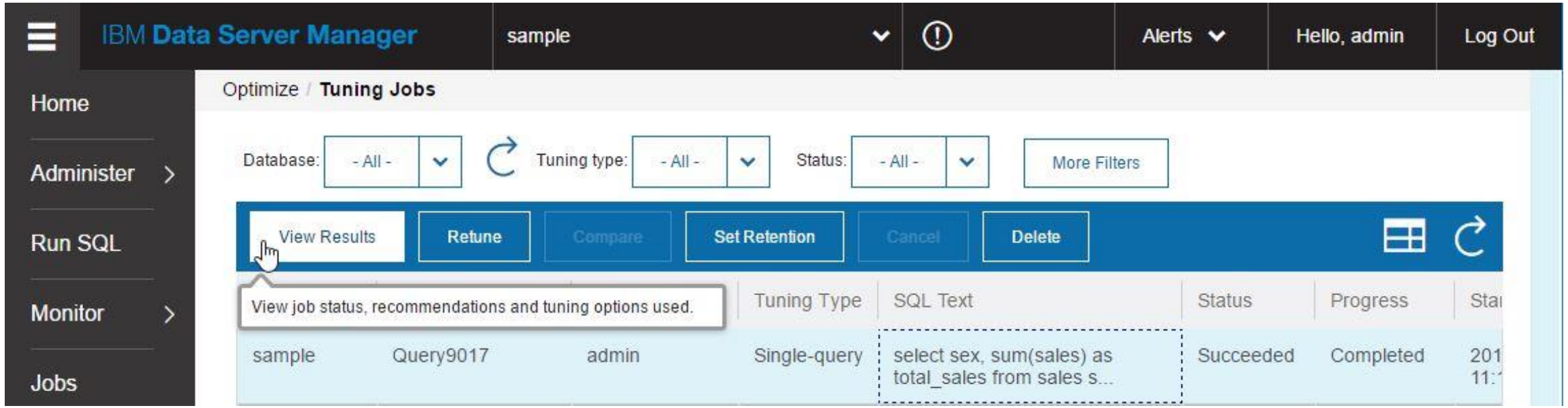





XTIVIA

# Tuning Queries with DSM Query Tuning (4)

7. Once tuning job has completed, click on it and then click “View Results”



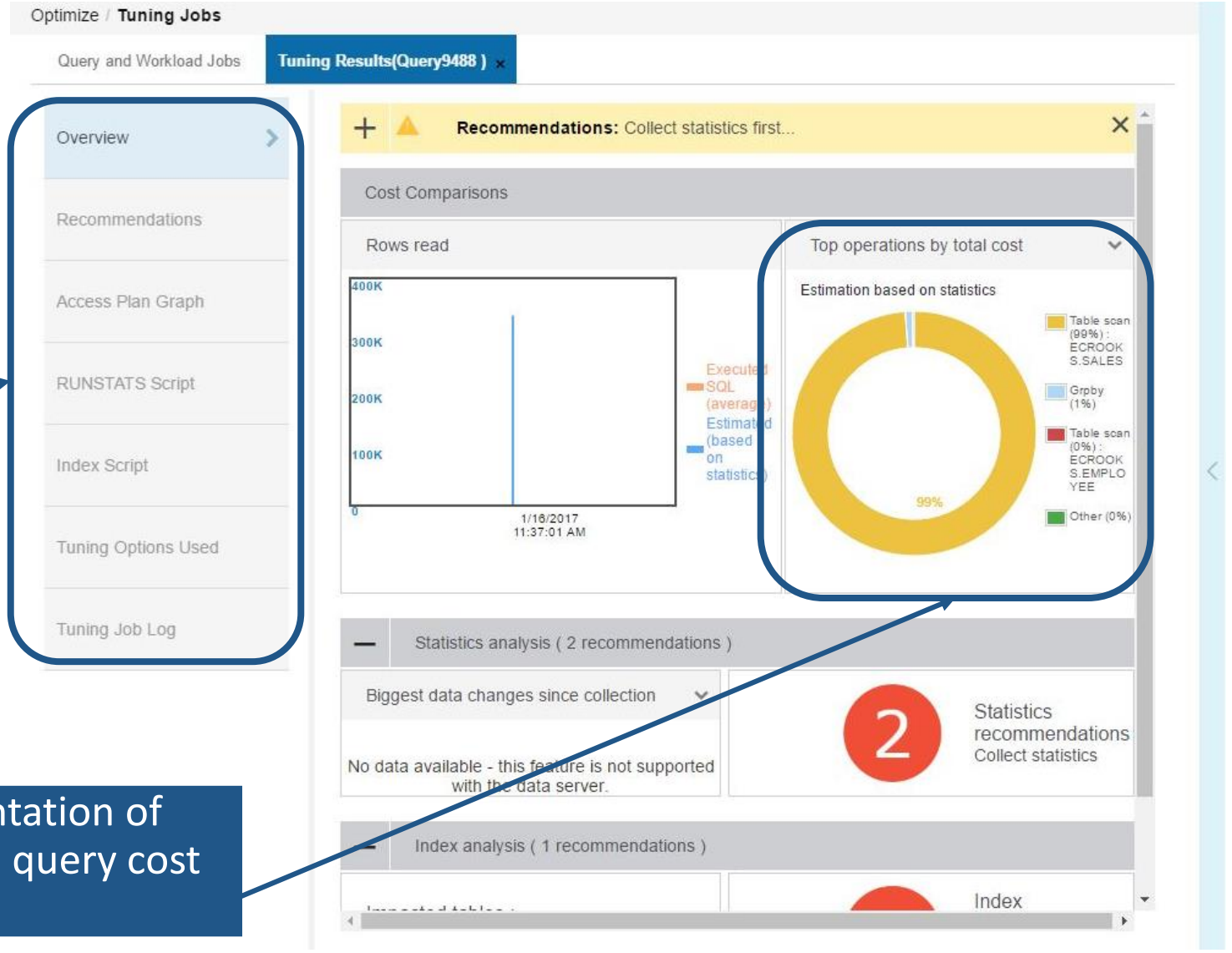



XTIVIA

# Query Tuning Output with DSM (1)

Menu to access the various information available

Graphical representation of which operators in query cost the most






XTIVIA

# Sections of DSM Tuning Results

- Overview
  - Cost comparisons with other SQL
  - Top operations by total cost (or other metrics)
  - Summary of statistics analysis
  - Summary of index analysis
- Recommendations
  - Very verbose description of just about everything, with links to more detail
- Access Plan Graph
  - Access Plan with the same details available as when you simply explain the SQL
- Runstats Script
  - Statements to collect statistics on the objects involved
- Index Script
  - Syntax to create indexes (option to test your own, and various combinations!)
- Tuning Options Used
  - Options specified when the query tuning job was executed
- Tuning Job Log
  - Simple list of basic information like start and stop times of the tuning job

# Unconventional Uses of Query Analysis Tools

# Unconventional Uses of Query Analysis Tools

1. Evaluating Impact of Multiple Recommended Indexes
2. Looking Through Queries from Locking Event Monitor for Indexing Opportunities

# Determining Which Recommended Indexes Provide Real Impact

- Design advisor does not tell you the relative impact of the different indexes recommended
- Some may have nearly no impact
- Design advisor tends to recommend index-only access, which leads to index bloat
- Consider the access priorities for the table

http://db2commerce.com/2015/06/24/multiple-index-regression-analysis/

# Compare Index Impact

- Recommend indexes on a query
- Collect high-level explain numbers as-is (using SQL is easiest)
- Add or subtract one or more indexes using the USE_INDEX column in the ADVISE_INDEX table
- Collect high-level explain numbers after using explain mode EVALUATE_INDEXES (using SQL is easiest)
- Repeat for each index or set of indexes that you want to evaluate

# Sample of Analysis of Multiple Indexes

| Before Cost | After Cost | | Potential difference | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 38772.4492 | 18741.7597 | | 20030.6895 | | | | | | | | |
| **Index Name** | **Tabsch** | **Tabname** | **fullkeycard** | **table card** | **uniquerule** | **exists** | **Timerons with this Index** | **Index impact addition** | **Timerons without this Index** | **Index impact subtraction** | **colnames** |
| IDX1505231647110 | DB2 | VENDOR_UNIT | 2976 | 2976 | U | N | 38772.4 | 0 | 18741.7597 | 0 | +VENDOR_UNIT_ID-VENDOR_ID |
| IDX1505231647160 | DB2 | LOCATION | 140445 | 140445 | U | N | 38772.4 | 0 | 18741.7597 | 0 | +LOCATION_ID-NAME |
| IDX1505231651490 | DB2 | ORDER | 642 | 52497 | U | N | 18766.3 | 20006.1075 | 38468.0898 | 19726.3301 | +TRAILER_SIZE+DL_AP_DATE2+CUSTOMER_RATE+E_ENTERED+COMMODITY+NEEDS_PAPERWORK+LOTAL_CHARGE+MILES+TERMINAL+SCAC+SEAL+PO_NUM+REFERENCE+WHO_ENTERED+ORDER_TYPEER_ID+DL_AT_TIME2+DL_AT_DATE2+DL_AT_TIME_AP_TIME2+DL_AP_TIME1+DL_AP_DATE1+PU_AT_2+PU_AT_TIME1+PU_AT_DATE1+PU_AP_TIME2+P_TIME1+PU_AP_DATE1+S_LOCATION_ID+CHASSISER+STATUS+ORDER_ID+C_LOCATION_ID+ACT_LOATION_ID |
| IDX1505231652450 | DB2 | RAIL_ETA | 649793 | 649793 | U | N | 38772.4 | 0 | 18741.7597 | 0 | +ORDER_ID-STATE |
| IDX1505231647170 | EDI | EDI_204_ADDITIONAL_FIELDS | 19897638 | 19897638 | D | N | 38772.4 | 0.0196 | 18741.7812 | 0.0215 | +REFERENCE+FIELD_NAME+INSERT_DT+FIELD_VA |
| IDX1505231648160 | DB2 | CODE | 5 | 520 | D | N | 38772.4 | 0 | 18741.7617 | 0.002 | +CODE_TEXT+TYPE+CODE_ID+CODE |
| IDX1505231648190 | DB2 | EDI_204 | 295885 | 1014841 | D | N | 38468.1 | 304.3399 | 18766.3183 | 24.5586 | +CUST_S_LOCATION_ID+ORDER_ID+REFERENCE |



XTIVIA

# Looking Through Queries from Locking Event Monitor for Indexing Opportunities

- This process assumes you're using one like this one (similar one gathering history also works): http://db2commerce.com/2012/01/23/analyzing-deadlocks-the-new-way/
- General process is:
  1. Create a workload based on SQL from the locking event monitor
  2. Generate explain and design advisor data
  3. Analyze data from ADVISE_WORKLOAD to identify SQL where the most advantage can be gained

# Create a Workload Based on Locking Event Monitor Data

```
insert into ecrooks.advise_workload(workload_name, statement_no, statement_text, frequency) (

select

        '20161011_locking2'

        , row_number() over()

        , rtrim(STMT_TEXT)

        , (select count(*) from ecrooks.lock_participant_activities lpa2 where
lpa.STMT_PKGCACHE_ID=lpa2.STMT_PKGCACHE_ID and xmlid in (select xmlid from ecrooks.lock_event where
event_timestamp > current timestamp - 7 days))

from ecrooks.lock_participant_activities lpa

where xmlid in (select xmlid from ecrooks.lock_event where event_timestamp > current timestamp - 7
days)

      )
```

- Workload name here is '20161011_locking2‘ – change as appropriate
- Output here is limited to the last 7 days – depending on what you have, you may want more or less

# Generate Explain and Advisor Data

```
db2advis -d sample -w 20161011_locking2 -keep -q db2inst1 > trans_3.advis
```

- Output to file is not the focus, but the tables this populates
- In my example, 65 indexes were recommended. I would not add all 65.
- This will give us the data to select SQL statements to do further analysis on

# Analyzing ADVISE_WORKLOAD

```
select decimal(cost_before,25,2) as before
        , decimal(cost_after,25,2) as after
        , frequency
        , decimal((cost_before-cost_after)*frequency,20,5) as weighted_diff
        , decimal(((cost_before-cost_after)*frequency)/(cost_before*frequency),5,2) as diff_pct
        , substr(statement_text,1,100) as stmt
from ecrooks.advise_workload
where cost_before is not null
        and workload_name='20161011_locking2'
order by weighted_diff desc
fetch first 20 rows only
with ur
```

# ADVISE_WORKLOAD Output

```
BEFORE                       AFTER                        FREQUENCY   WEIGHTED_DIFF          DIFF_PCT STMT
---------------------------- ---------------------------- ----------- ---------------------- -------- -----------------------------------
           273136630000.00              271870330000.00           6       7597800000.00000     0.00 DELETE FROM (SELECT 1 FROM MEMBER WHE
               72467712.00                   306432.00          24       1731870720.00000     0.99 select process_state, count(*) from X
                1317120.00                   107520.00         840       1016064000.00000     0.91 SELECT CASE WHEN ((SELECT COUNT(*) fr
               40763088.00                   172368.00          18        730632960.00000     0.99 select process_state, count(*) from X
           287441850000.00              287263060000.00           4        715160000.00000     0.00 DELETE FROM (SELECT 1 FROM MEMBER WHE
                2822400.00                  2069760.00         840        632217600.00000     0.26 SELECT CASE WHEN ((SELECT COUNT(*) AS
               35014896.00                  2690928.00          12        387887616.00000     0.92 select count (*) from x_xorderitemsta
(
select x.x_xorderitemstate_i
               34266096.00                  2688480.00          12        378931392.00000     0.92 select count(*) from x_xorderitemstat
                1197840.00                   781200.00         840        349977600.00000     0.34 SET :HV00023  :HI00023  = (SELECT SUB
               18116640.00                   623088.00          12        209922624.00000     0.96 select * from x_xorderitemstate where
                 120960.00                    13440.00         840         90316800.00000     0.88 SELECT CASE WHEN ( (SELECT COUNT(*) F
                 114240.00                    13440.00         840         84672000.00000     0.88 SELECT CASE WHEN ((SELECT COUNT(*) fr
                8753724.00                   672732.00           6         48485952.00000     0.92 select count(*) from x_xorderitemstat
```

- First statement low % improvement – I have a rewrite that does better for that
- Weighted difference is only given the frequency this statement is caught in locking issues
- Carriage returns sometimes complicate output (CLPPLUS may help)
- SQL here is truncated to present better – can return full statement with another query
- Some queries here very similar

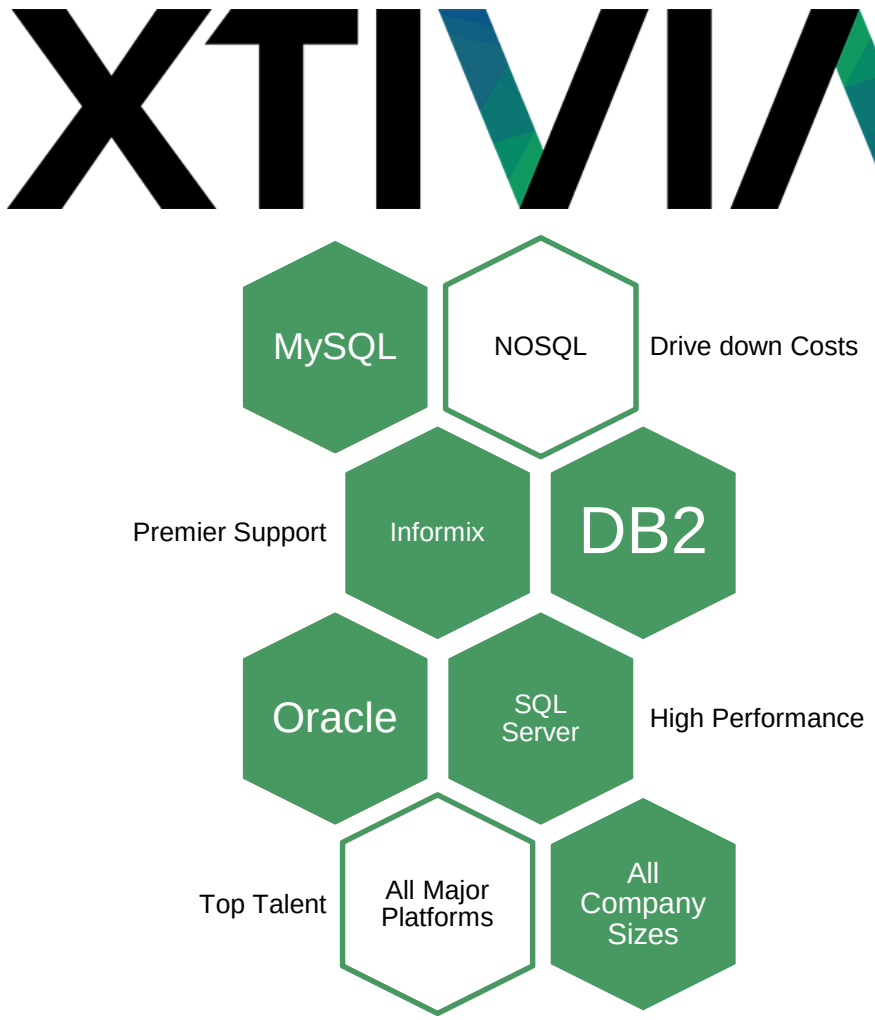
# Ember Crooks

XTIVIA
http://www.virtual-dba.com/
http://www.xtivia.com/

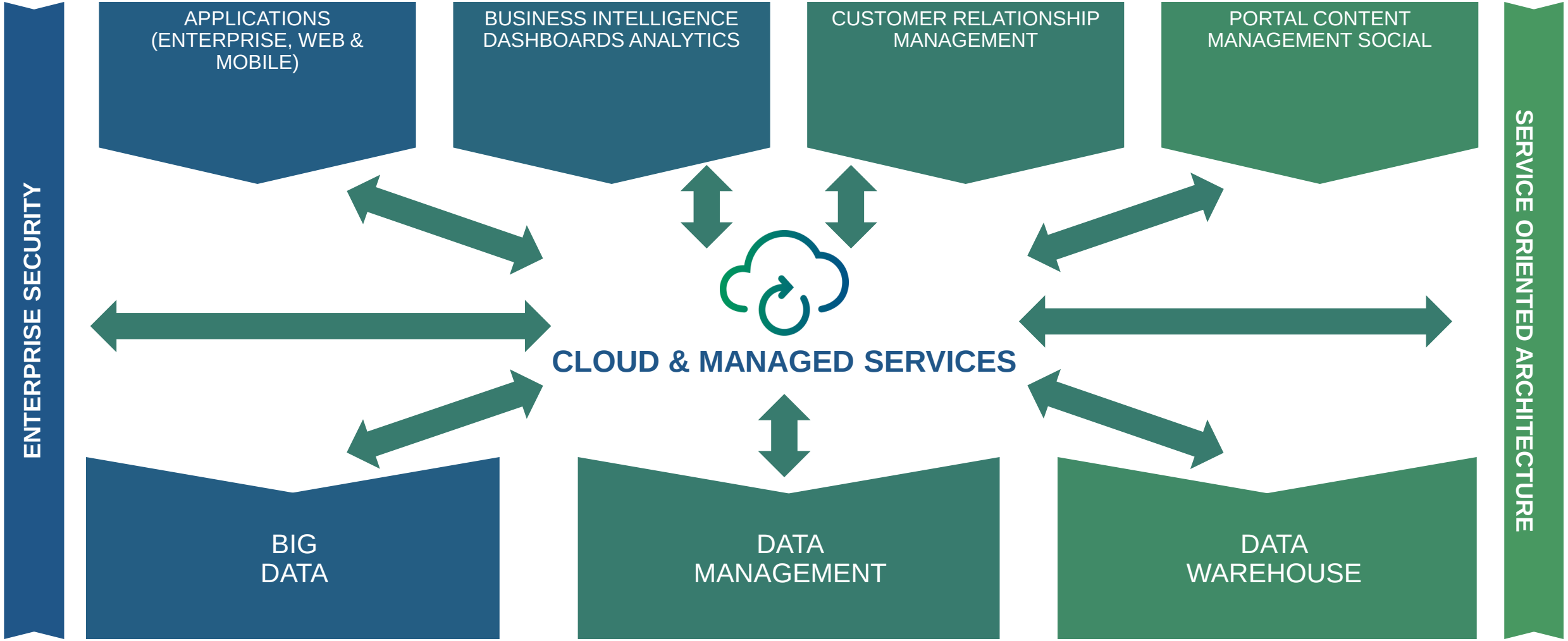
ember.crooks@gmail.com

Blog: http://db2commerce.com

Twitter: @ember_crooks



# INDUSTRY LEADING TALENT – CONSISTENT RESULTS – PROVEN METHODOLOGY






XTIVIA