#IDUGDb2

SQL Coding Best Practices

Rotterdam, Netherlands 2019

Phil Gunning

Gunning Technology Solutions, LLC

Session code: F12

October 23, 2019 1440-1540

LUW

IDUG
Leading the Db2 User Community since 1988

Objectives

- Learn and understand DB2 Predicate Rules
- Understand range delimiting predicates and index usage
- Learn how to Explain SQL and Interpret the results
- Understand how Generated columns, Multidimensional Clustering and Function based indexes can be used to improve performance

Outline

- DB2 Predicate Rules
- Index Usage
- Jump Scan
- Db2 Explain, Db2exfmt and other Visual Explain tooling
- Suboptimal SQL examples from actual scenarios

Outline

- SQL rewrite tips and techniques
- SQL Rewrite solutions
- More tips and techniques
  - MDC
  - Generated Columns

Suboptimal SQL

- Suboptimal SQL results in lost revenue and lost business opportunities
- Can destroy the business
  - Example: Can't cut employee paychecks or pay vendors
- If Web-facing, customers don't come back
- If internal, causes strife within departments
- Lost productivity
- Increased resource consumption
- Lost customers (external and internal)
- Damaged business reputation
  - Bad press

Top Causes of Suboptimal SQL

- SQL Training shortfall for developers
- Improper Index Design
  - Many times not much thought given to designing indexes
  - All kinds of crazy designs seen like an index on every column in a table
  - Lack of logical design or DBA involvement
  - Data Modeling decaying skill
    - Not emphasized very much anymore
- Explain not used or Developers do not have access to it
- Data modelling (logical design) not in-sync with continuous delivery

Characteristics of Suboptimal SQL

- Join predicates missing or not indexed
- Local predicates (those in the select list) not indexed for potential index-only access
- Order by predicates not indexed or indexes not created with “ALLOW REVERSE SCANS”
  - Note “ALLOW REVERSE SCANS” now default in DB2 9.5
- Foreign key indexes not defined
  - Note that EXPLAIN enhanced in DB2 9.5 to show use of FK (RI)
- Misunderstanding of IXSCAN operator

Characteristics of Suboptimal SQL

- DB2 built-in functions such as UCASE causing IXSCAN of entire index
  - Generated column
- Isolation level not understood
  - WITH UR
- Company culture does not allow time for explain of SQL before it goes into production
  - Nowadays, this is very prevalent
- Developers not aware of explain capabilities and options
- Design Advisor not used or misinterpreted

Characteristics of Suboptimal SQL

- High number of rows read
- Physical IO
- Long running SQL
- High timeron cost
- High number of logical reads and high CPU usage
- Concurrency problems
  - Unordinary amount of lock timeouts or deadlocks

Classes of Predicates*

- Range Delimiting
- Index SARGable
- Data SARGable
- Residual

* Ranked best to worst

Predicate Example Index

- For the following predicate rule examples, assume that an index has been created on Col A, Col B, and Col C Asc as follows:
  - ACCT_INDX:

Col A	Col B	Col C

Predicates

- Range Delimiting
  - Used to bracket an index scan
  - Uses start and stop predicates
  - Evaluated by the Index Manager
- Easily confirmed via Explain!

Range Delimiting Example

Col A = 3 and Col B = 6 and Col C = 8	In this case the equality predicates on all the columns of the index can be applied as start-stop keys and they are all range delimiting

Col A	Col B	Col C

Predicates

- Index SARGable
  - Are not used to bracket an index scan
  - Can be evaluated from the index if one is chosen
  - Evaluated by the Index Manager

Index SARGable Example

Col A = 9 and Col C = 4	Col A can be used as a range delimiting (start-stop) predicate. Col C can be used as an Index SARGable predicate, it cannot be used as a range delimiting since there is no predicate on Col B. Starting with columns in the index, from left to right, the first inequality predicate stops the column matching.

Col A	Col B	Col C

Index SARGable Example

Col D = 9, Col E=8 and Col C = 4	Col D and E cannot be used as range-delimiting and are also not present in the index. Col C can be used as an Index SARGable predicate, it cannot be used as a range delimiting since there is no predicate on Col A or Col B.

Col A	Col B	Col C

INDEX SCAN OF ENTIRE INDEX!

Predicates

- Data SARGable
  - Cannot be evaluated by the Index Manager
  - Evaluated by Data Management Services
- Requires the access of individual rows from the base table

Data SARGable Example

Col A = 3 and Col B <= 6 and Col D = 9	Col A is used as a start-stop predicate, Col B is used as a stop predicate, and Col D which is not present in the index is applied as a Data SARGable predicate during the FETCH from the table

Col A	Col B	Col C

Residual Predicates

- Residual Predicates
  - Cannot be evaluated by the Index Manager
  - Cannot be evaluated by Data Management Services
- Require IO beyond accessing the base table
- Predicates such as those using quantified sub-queries (ANY, ALL, SOME, or IN), LONG VARCHAR, or LOB data
- Correlated Sub-queries
- Are evaluated by Relational Data Services and are the most expensive type of predicates

Residual Predicate Example

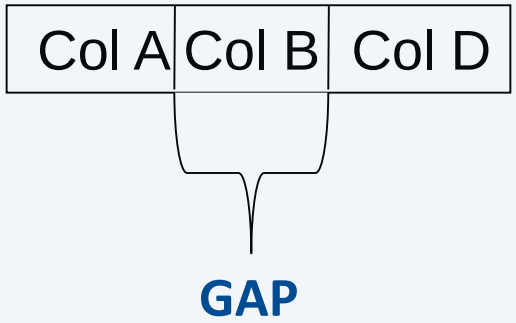
Col B = 4 and UDF with external action(Col D)	In this case the leading Col A does not have a predicate. Col B can only be used as an Index SARGable predicate (where the whole index is scanned). Col D involves a user defined function which will be applied as a residual predicate

Jump Scan

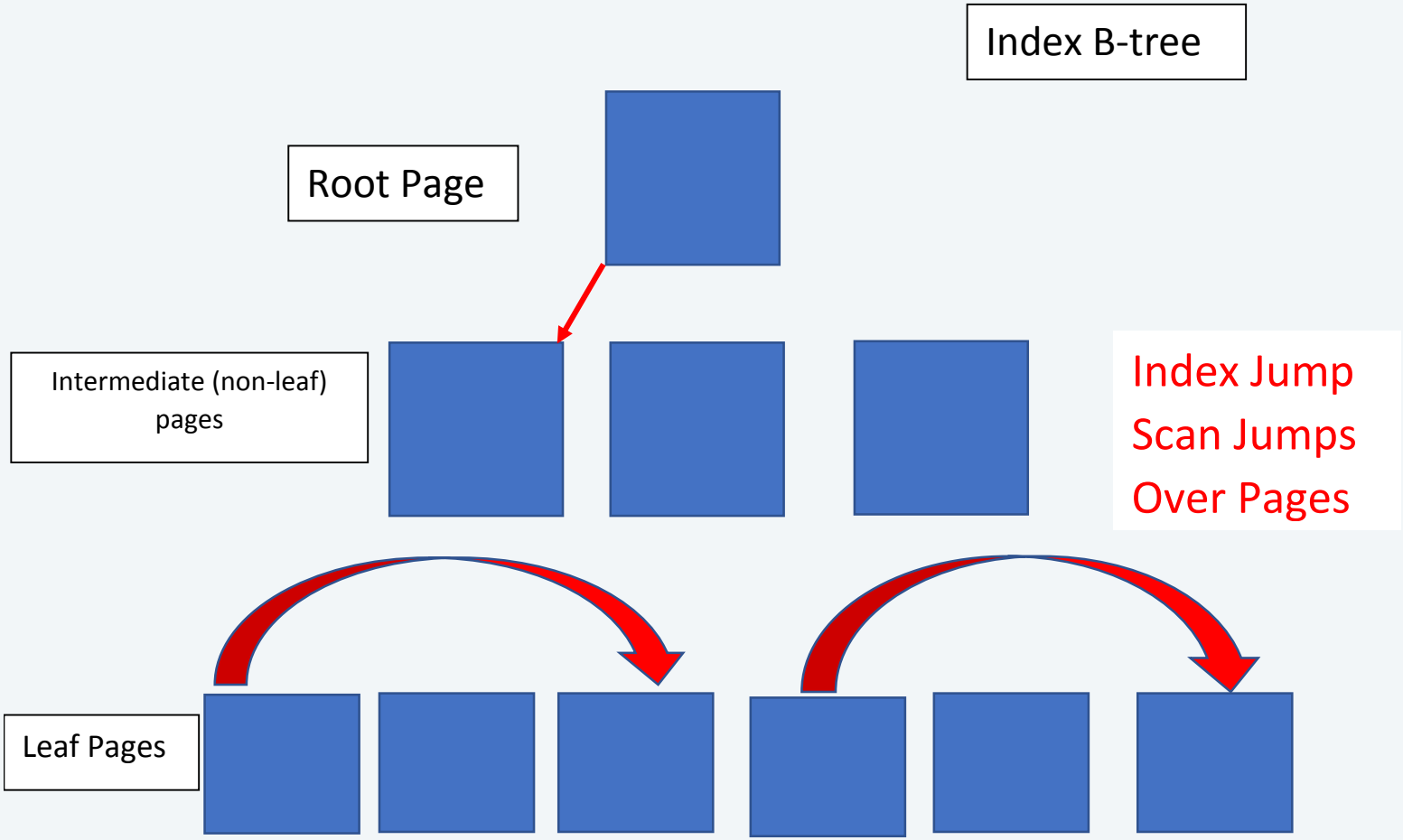
- Introduced in Db2 10.1
- Enables jumping over leaf (gaps) pages
- Can result in fewer indexes and some performance improvement
- Your Mileage May Vary (YMMV)

Jump Scan Example

Col A = 3 and Col D = 9	Col A is used as a start-stop predicate, Col B is not specified, and Col D is specified and contained in the index. Since COL B is not specified in the query it creates a GAP. With JUMP SCAN the optimizer may elect to use a JUMP SCAN and use the index.



Jump Scan – How it Works



Monitoring Jump Scans

- MON_GET_INDEX table function
  - DB CFG Parameter MON_OBJ_METRICS must be set to BASE or Extended
- MON_GET_INDEX table function INDEX_JUMP_SCANS monitoring element
- Shown on Explain output if used

ADDITIONAL REGISTRY SETTINGS

- DB2_SKIPINSERTED=YES
- DB2_INLST_NLJN=YES *
- DB2_ANTIJOIN=EXTEND*
- DB2_REDUCED_OPTIMIZATION*
- DB2_EVALUNCOMMITTED=YES
- DB2_MINIMIZE_LISTPREFETCH*
- DB2_WORKLOAD=SAP, etc

ISOLATION LEVELS

- Default of CURSOR STABILITY (CS) usually a good choice
- UNCOMMITTED READ (UR) for read only and reporting

Review of Isolation Levels

Comparison of isolation levels				
	UR	CS	RS	RR
Can an application see uncommitted changes made by other application processes?	Yes	No	No	No
Can an application update uncommitted changes made by other application processes	No	No	No	No
Can the re-execution of a statement be affected by other application processes? 1	Yes	Yes	Yes	No 2
Can updated rows be updated by other application processes? 3	No	No	No	No
Can updated rows be read by other application processes that are running at an isolat	No	No	No	No
Can updated rows be read by other application processes that are running at the UR is	Yes	Yes	Yes	Yes
Can accessed rows be updated by other application processes? 4	Yes	Yes	No	No
Can accessed rows be read by other application processes?	Yes	Yes	Yes	Yes
Can the current row be updated or deleted by other application processes? 5	Yes/No 6	Yes/No 6	No	No

TUNING TIP -- MONREPORT PKGCACHE Executable ID

- Call the MONREPORT reporting module
  - Db2 “call monreport.pkgcache”
  - Review the top 10 SQL based on your most important criteria
  - Identify the Executable ID
- Pass the Executable ID to the Explain_From_Section stored procedure
- Format the SQL with db2exfmt

Predicate Best Practices

- Use Range Delimiting predicates whenever possible
- Verify via your favorite form of Explain
  - Visual Explain
  - db2exfmt
  - Third party vendor tool

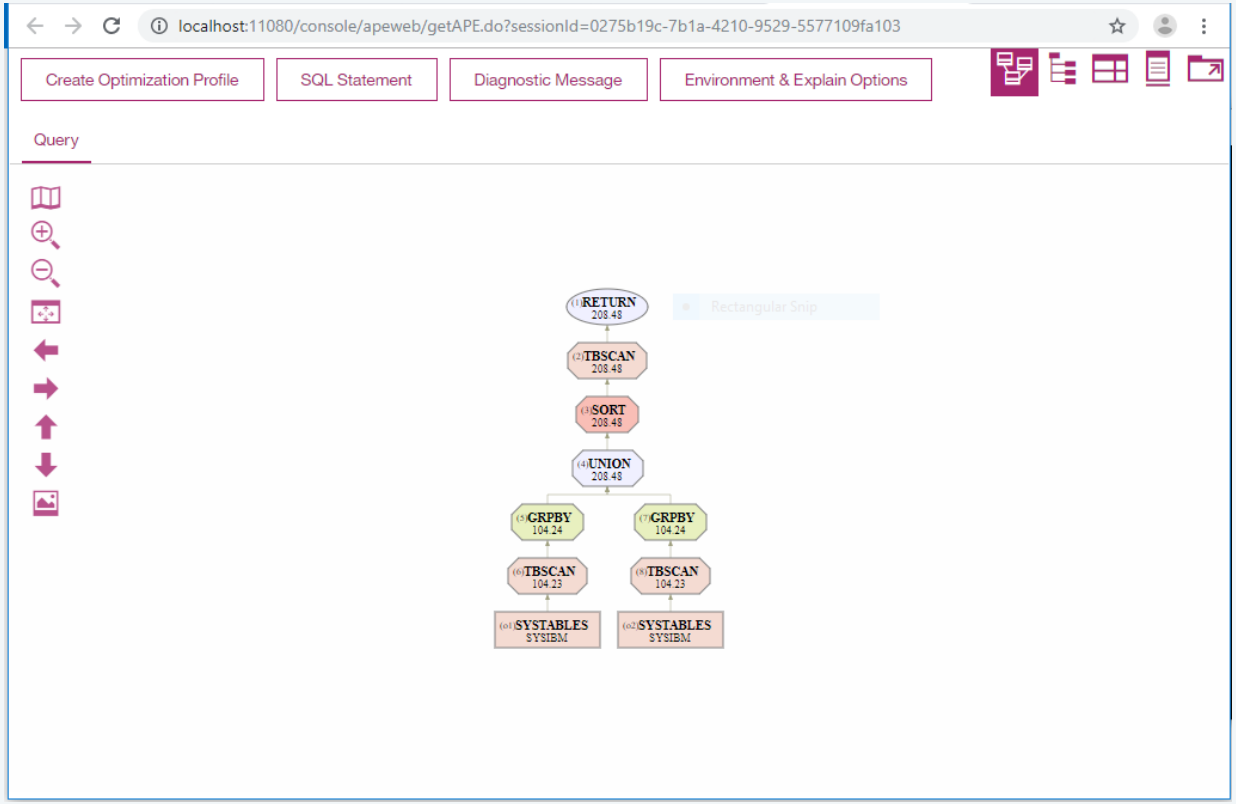
Predicate Best Practices

- Use Range Delimiting predicates whenever possible
- Verify via your favorite form of Explain
  - Visual Explain
  - db2exfmt
  - Third party vendor tool

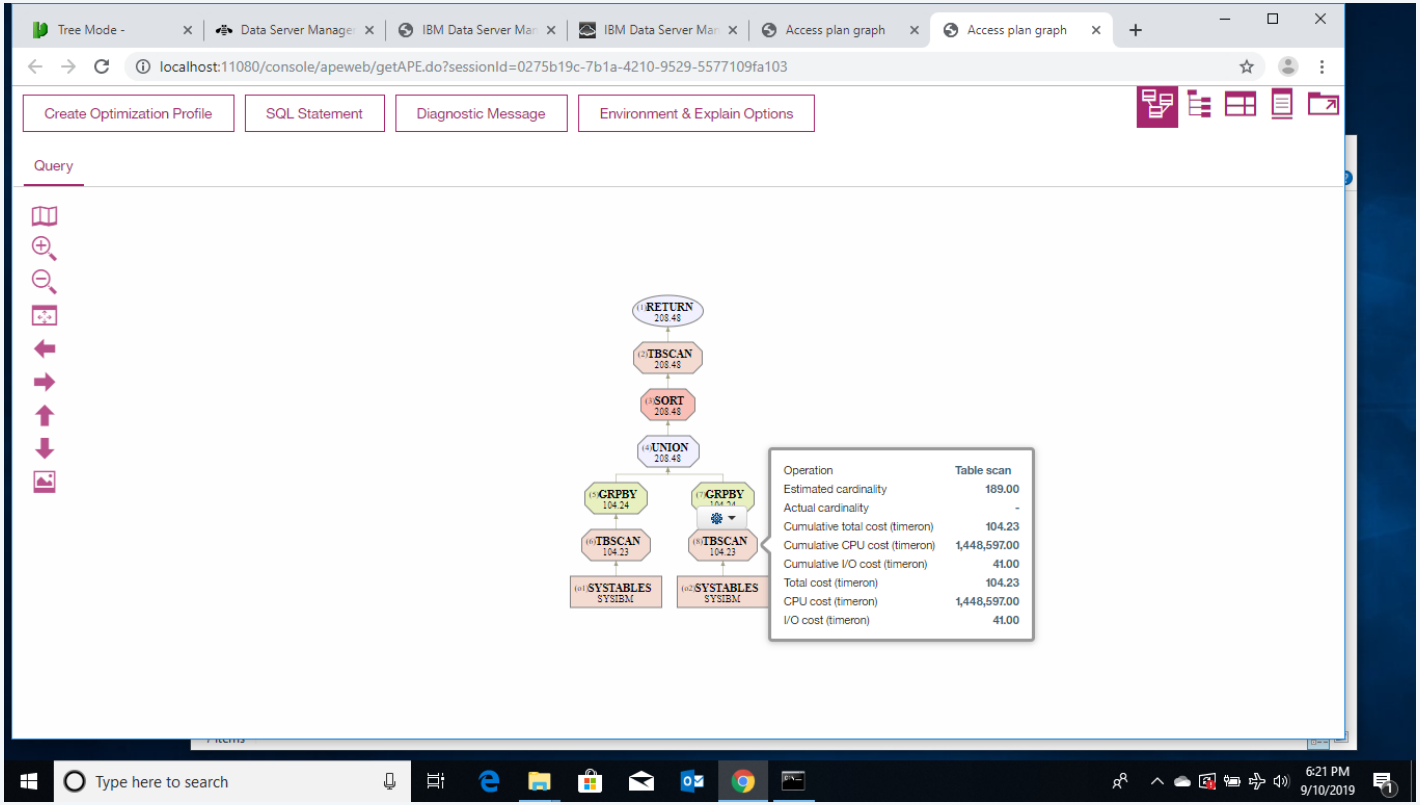
EXPLAIN

- Verify SQL access paths with your favorite form of Explain
  - Db2exfmt (my favorite and supports favorite)
  - Data Server Manager visual explain
  - Third party vendor tool visual explain
- Look for table scans, sorts, temp table usage and index RIDLIST usage
- Rewrite the SQL if possible
- Try to eliminate these with an index redesign or new index if possible and there are not too many indexes on the table(s) being used

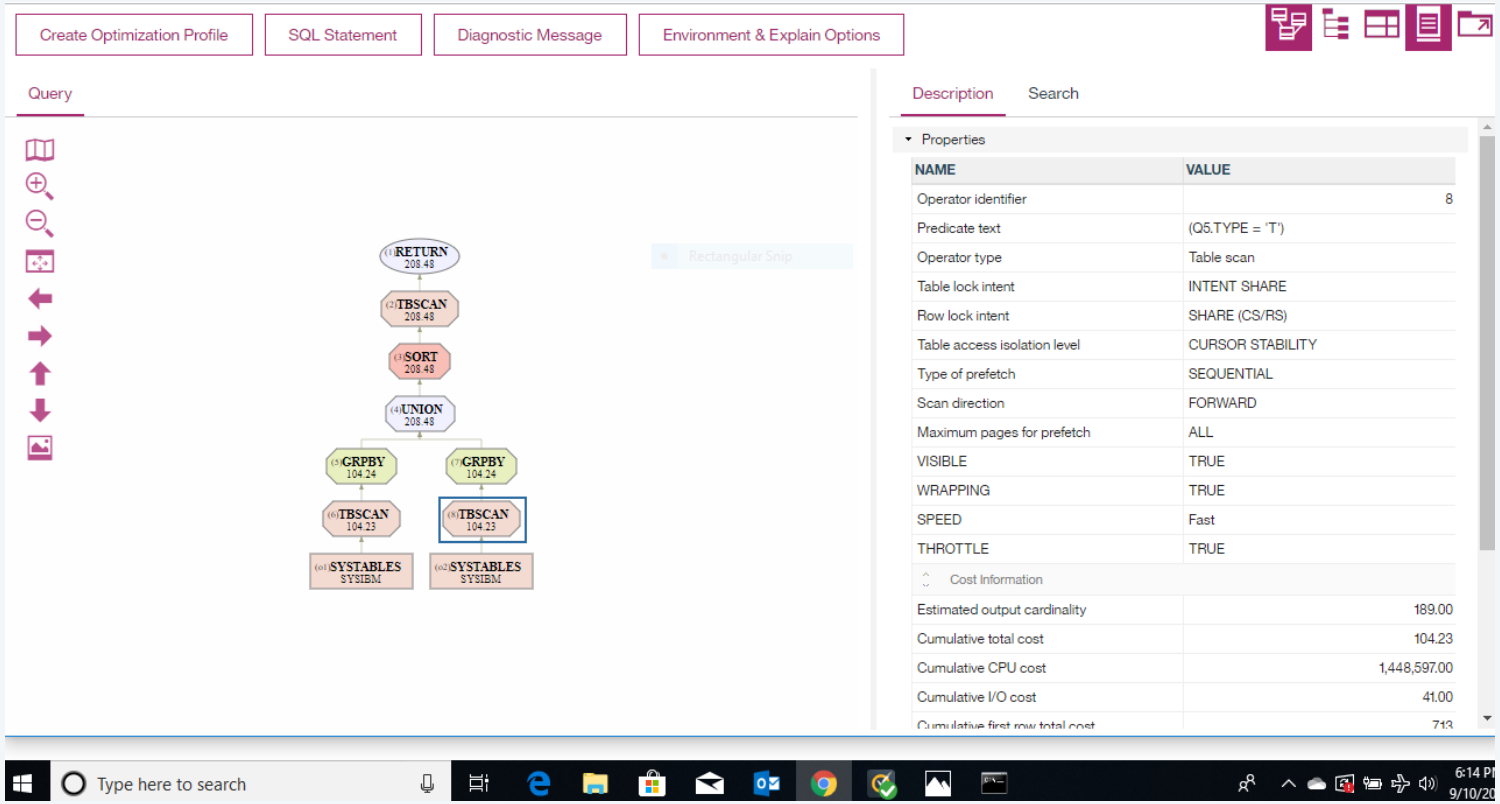
Data Server Manager Explain



Data Server Manager Explain



Data Server Manager Explain



db2exfmt

- Command line tool to format EXPLAIN tables
- Produces an access path tree along with optimized SQL and details on operations and objects involved
- Preferred format when working with Db2 support

db2exfmt

- Set the current explain mode to explain
- Verify that explain mode is set
- Executed the query to be Explained
- Invoke db2exfmt from the command line
- Take the defaults and specify and output file when prompted

Invoking db2exfmt

```
c:\Users\Administrator>db2 set current explain mode explain
DB20000I  The SQL command completed successfully.

c:\Users\Administrator>db2 "select * from sysibm.sysdummy1"
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604

c:\Users\Administrator>db2 -tvf te2.txt
SELECT COUNT(*)   FROM db2admin.ACCOUNT_MACHINE, db2admin.CLIENT_ACC WHERE db2admin.ACCOUNT_MACHINE.ACC_NUM = db2admin.CLIENT_ACC.ACC_NUM AND HEX(MACHINE_ID) =? aND db
min.CLIENT_ACC.CASINO_ID = ?
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604


c:\Users\Administrator>db2exfmt
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

Enter Database Name ==> migamprd
Connecting to the Database.
Connect to Database Successful.
Enter up to 26 character Explain timestamp (Default -1) ==>
Enter up to 128 character source name (SOURCE_NAME, Default %%) ==>
Enter source schema (SOURCE_SCHEMA, Default %%) ==>
Enter section number (0 for all, Default 0) ==>
Enter outfile name. Default is to terminal ==> te2ux.txt
Output is in te2ux.txt.
Executing Connect Reset -- Connect Reset was Successful.

c:\Users\Administrator>
```

db2exfmt

- Shows the optimizer query access plan as a tree
- Leaf nodes are data
- Internal nodes are operators such as join, sort, filter, group, etc

```
Total Cost:                 133834
          Query Degree:            1
          Rows
          RETURN
          (   1)
          Cost
           I/O
           |
            1
          GRPBY
          (   2)
          133834
          125332
           |
          62341.2
          HSJOIN^
          (   3)
          133830
          125332
      /------+-------\
   496142          436388
   IXSCAN          IXSCAN
   (   4)          (   5)
   4394.55         129359
   4054.58         121277
     |               |
 3.47299e+006     1.09097e+007
INDEX: PHIL      INDEX: DB2ADMIN
GTSX_CLIENT_ACC_XG_ACCOUNT_MACHINE
```

Db2exfmt Query Graph

```
Total Cost:                 133834
Query Degree:                    1
Rows
RETURN
(   1)
Cost
I/O
|
1
GRPBY
(   2)
133834
125332
|
62341.2
HSJOIN^
(   3)
133830
125332
/------+-------\
496142          436388
IXSCAN          IXSCAN
(   4)          (   5)
4394.55         129359
4054.58         121277
|               |
3.47299e+006    1.09097e+007
```

Total cost of the query in timerons

Name of operator and number of operator in the details section Number of rows expected
Cumulative Cost in timerons
Numbrer of I/Os that have been done

3) HSJOIN Operator db2exfmt Details

3) HSJOIN: (Hash Join)

Cumulative Total Cost:	133830
Cumulative CPU Cost:	3.03806e+010
Cumulative I/O Cost:	125332
Cumulative Re-Total Cost:	133830
Cumulative Re-CPU Cost:	3.03806e+010
Cumulative Re-I/O Cost:	125332
Cumulative First Row Cost:	133830
Estimated Bufferpool Buffers:	4054.71

5) IXSCAN Operator db2exfmt Details

```
5) IXSCAN: (Index Scan)
        Cumulative Total Cost:          129359
        Cumulative CPU Cost:            2.92343e+010
        Cumulative I/O Cost:            121277
        Cumulative Re-Total Cost:       8072.91
        Cumulative Re-CPU Cost:         2.84853e+010
        Cumulative Re-I/O Cost:         0
        Cumulative First Row Cost:      22.9872
        Estimated Bufferpool Buffers:   121278

        Arguments:
        ---------
        CUR_COMM: (Currently Committed)
                TRUE
        JN INPUT: (Join input leg)
                INNER
        LCKAVOID: (Lock Avoidance)
                TRUE
        MAXPAGES: (Maximum pages for prefetch)
                118850
        PREFETCH: (Type of Prefetch)
                SEQUENTIAL,READAHEAD
        ROWLOCK : (Row Lock intent)
                SHARE (CS/RS)
        SCANDIR : (Scan Direction)
                FORWARD
        SKIP_INS: (Skip Inserted Rows)
                TRUE
```

5)IXSCAN Operator db2exfmt Details

```
Predicates:                    ----------
                    4) Sargable Predicate,
                            Comparison Operator:              Equal (=)
                            Subquery Input Required:  No
                            Filter Factor:                    0.04
                            Predicate Text:
                            --------------
                            (HEX(Q2.MACHINE_ID) = ?)
                    Input Streams:
                    -------------
                            3) From Object DB2ADMIN.XG_ACCOUNT_MACHINE
                                    Estimated number of rows:         1.09097e+007
                                    Number of columns:                3
                                    Subquery predicate ID:            Not Applicable
                                    Column Names:
                                    ------------
                                    +Q2.$RID$+Q2.MACHINE_ID+Q2.ACC_NUM
                    Output Streams:
                    --------------
                            4) To Operator #3
                                    Estimated number of rows:         436388
                                    Number of columns:                1
                                    Subquery predicate ID:            Not Applicable
                                    Column Names:
                                    ------------
                                    +Q2.ACC_NUM
```

A Word About Filter Factors

- The filter factor for the predicate is shown in Operator #5 of the db2exfmt

```
4) Sargable Predicate,
                    Comparison Operator:                    Equal (=)
                    Subquery Input Required:        No
                    Filter Factor:                          0.04
                    Predicate Text:
                    --------------
                    (HEX(Q2.MACHINE_ID) = ?)
          Input Streams:
          -------------
                    3) From Object DB2ADMIN.XG_ACCOUNT_MACHINE
                              Estimated number of rows:     1.09097e+007
                              Number of columns:            3
                              Subquery predicate ID:                Not Applicable
                              Column Names:
                              ------------
                              +Q2.$RID$+Q2.MACHINE_ID+Q2.ACC_NUM
          Output Streams:
          --------------
                    4) To Operator #3
                              Estimated number of rows:     436388
                              Number of columns:            1
                              Subquery predicate ID:                Not Applicable
```

Identifying Suboptimal SQL

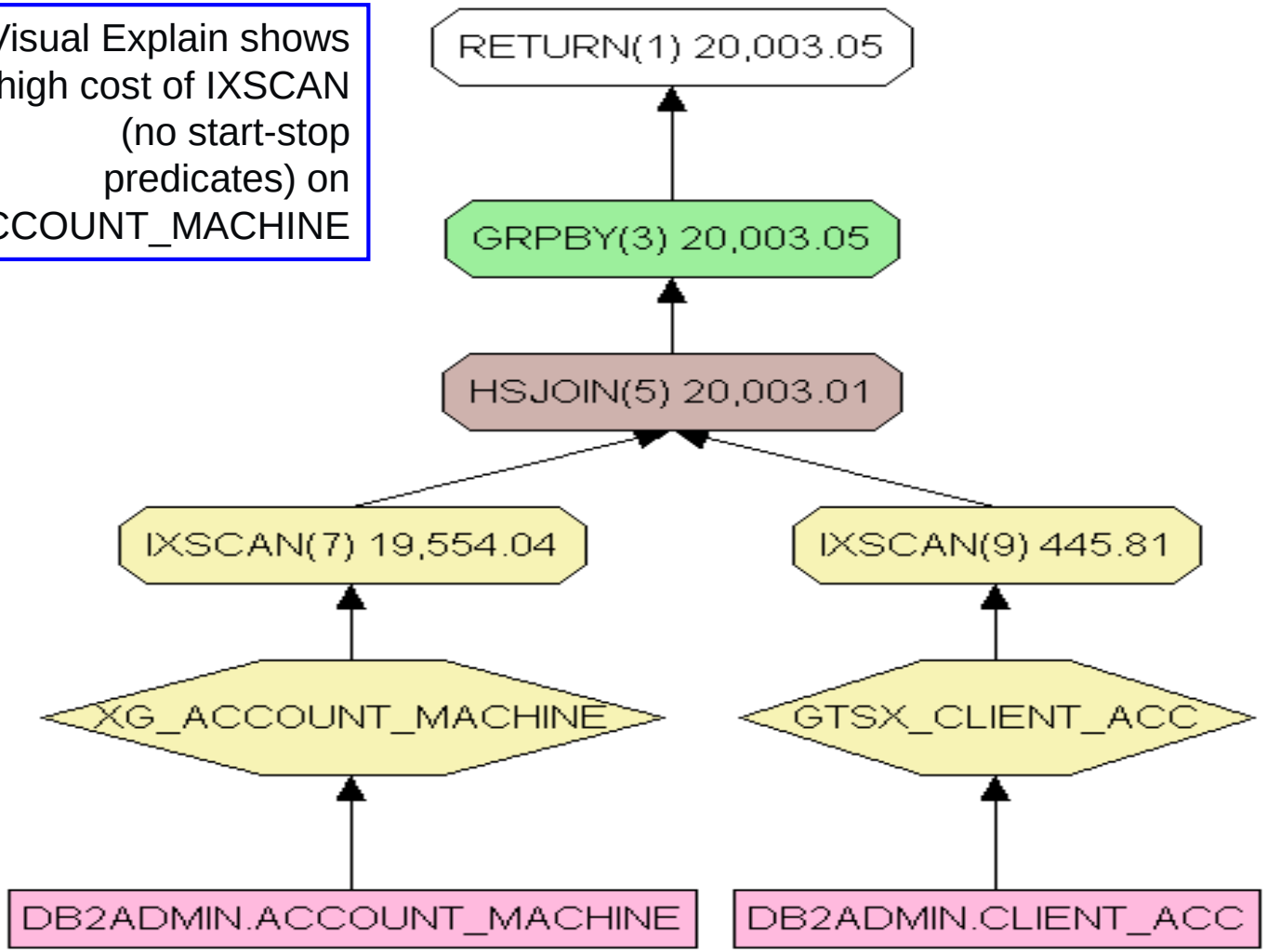
- MONREPORT Reporting module
- MON_GET Table Functions
- Top 10 Query
- Db2pd
- Event Monitors
- Third Party Vendor Tool
- Combination of TOP or TOPAS and db2pd/PID cross-reference to MON_GET functions

Another Suboptimal SQL Query

```
SELECT COUNT(*)
FROM   account_machine,
       client_acc
WHERE  account_machine.acc_num = client_acc.acc_num
       AND Hex(machine_id) = ?
       AND client_acc.casino_id = ?
```

IDENTIFIED via Top 10 SQL Query

Visual Explain shows high cost of IXSCAN (no start-stop predicates) on ACCOUNT_MACHINE



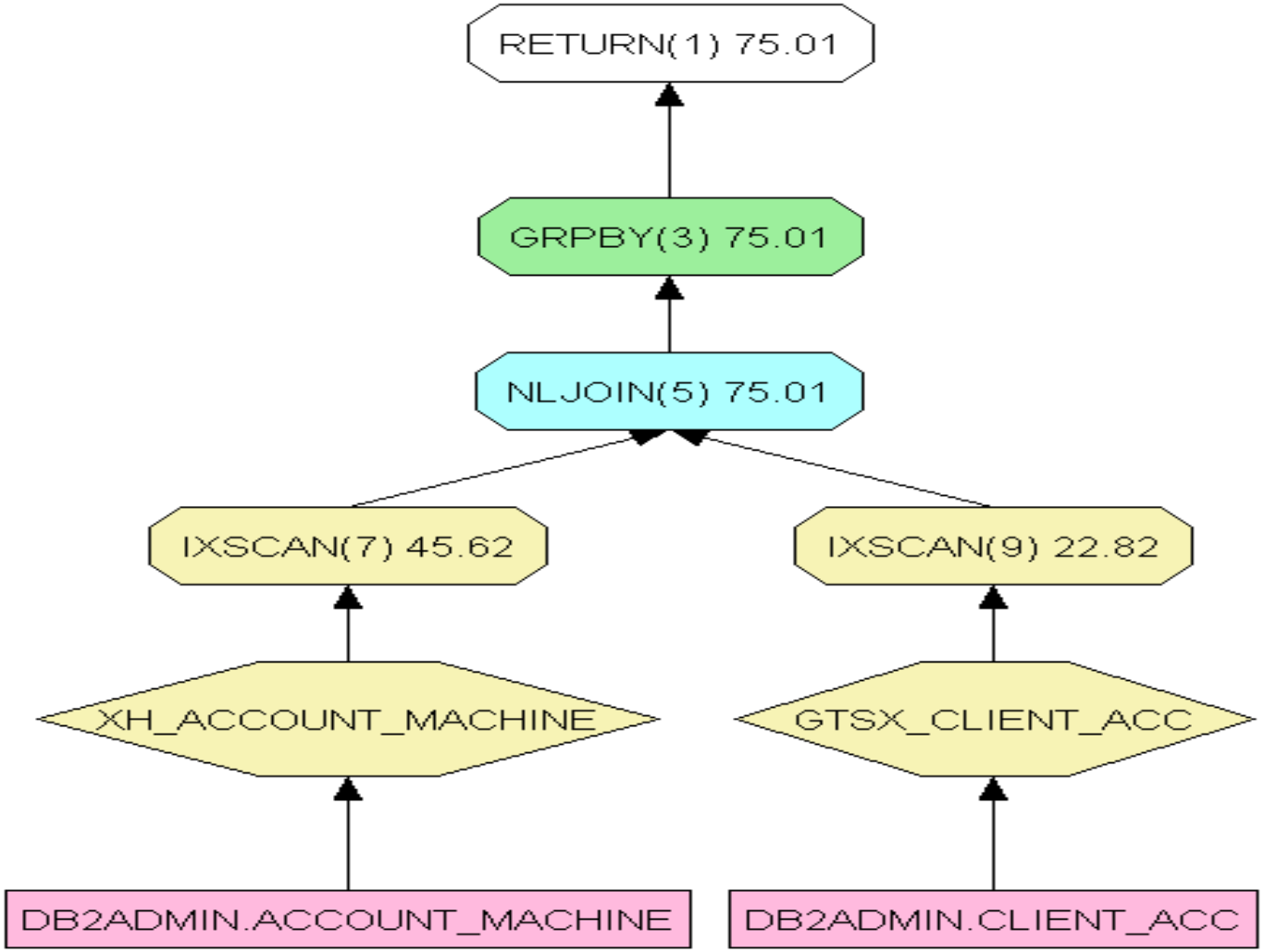
Solution was to Create a Generated Column

1. SET INTEGRITY FOR DB2ADMIN.ACCOUNT_MACHINE OFF;

2. ALTER TABLE DB2ADMIN.ACCOUNT_MACHINE ADD COLUMN MACHINE_HEX_ID CHARACTER (127) NOT NULL GENERATED ALWAYS AS (HEX(MACHINE_ID));

3. SET INTEGRITY FOR DB2ADMIN.ACCOUNT_MACHINE IMMEDIATE CHECKED FORCE GENERATED;

4. CREATE INDEX DB2ADMIN.XH_ACCOUNT_MACHINE ON DB2ADMIN.ACCOUNT_MACHINE (MACHINE_HEX_ID, ACC_NUM) ALLOW REVERSE SCANS;


RETURN(1) 75.01
GRPBY(3) 75.01
NLJOIN(5) 75.01
IXSCAN(7) 45.62
IXSCAN(9) 22.82
XH_ACCOUNT_MACHINE
GTSX_CLIENT_ACC
DB2ADMIN.ACCOUNT_MACHINE
DB2ADMIN.CLIENT_ACC

Expression-based Index Introduced in DB2 10.5

- Allows creation of indexes based on an expression
- Online REORG not supported
- Refer to RFI - https://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=114749

Advanced search | My searches | Top requests | Browse requests | View request

The product development team will review the request and provide status updates as decisions are made regarding the request. You can send an email to inform others of the request.

A key icon indicates that the field is displayed only to the original submitter. The key icon next to a request indicates that the request is a private request.

Headline:	Inplace Reorg should be allowed on a table with an Expression Based Index
ID:	114749

Details | Comments | Attachments | Reconsideration | Associated requests

RTC ID:	48547
State:	Open
Status:	Uncommitted Candidate
Created on:	01 Jan 2018, 11:40 PM Eastern Time (ET)
Updated on:	22 Sep 2019, 01:42 PM Eastern Time (ET)

Suboptimal SQL MDC Candidate Query

```
WITH INFO AS
    (SELECT PLAYERNAME AS PLAYERNAME, ACC_NUM AS ACC_NUM,STAKE AS STAKE,
    PLAY_COUNT AS PLAY_COUNT,GAME_ID AS GAME_ID,NUM_QUALIFY AS NUM_QUALIFY
    , REAL_PRIZE_PAID AS REAL_PRIZE_PAID, REBUY_COUNT AS REBUY_COUNT,
    PROMO_PRIZE_PAID AS PROMO_PRIZE_PAID , RANK() OVER (ORDER BY STAKE
    DESC) AS PLAYER_RANK
    FROM db2admin.CT_PLAYER
    WHERE TOURNAMENT_ID = ? AND NUM_QUALIFY=0
    UNION
    SELECT PLAYERNAME AS PLAYERNAME, ACC_NUM AS ACC_NUM,STAKE AS STAKE,
    PLAY_COUNT AS PLAY_COUNT, GAME_ID AS GAME_ID, NUM_QUALIFY AS
    NUM_QUALIFY , REAL_PRIZE_PAID AS REAL_PRIZE_PAID, REBUY_COUNT AS
    REBUY_COUNT, PROMO_PRIZE_PAID AS PROMO_PRIZE_PAID, 0 AS PLAYER_RANK
    FROM db2admin.CT_PLAYER
    WHERE TOURNAMENT_ID = ? AND ACC_NUM IN ('EH0144300844','GP0805174740',
    'GP0280683162','SL0763326234','GP0806937257','SL0410586631',
    'SL0871800961','GP0002320186','GP0006520691','SD0580234716',
    'SL0919369066','SL0673693302','EH0131748166','HT0232729921',
    'GP0550097653','GP0695261884','EP0939931413','EF0273763788',
    'GP0035242171','GP0994999656','SL0237577932','EH0109845675'))
SELECT *
FROM INFO
WHERE ACC_NUM IN ('EH0144300844','GP0805174740','GP0280683162','SL0763326234',
     'GP0806937257','SL0410586631','SL0871800961','GP0002320186',
     'GP0006520691','SD0580234716','SL0919369066','SL0673693302',
     'EH0131748166','HT0232729921','GP0550097653','GP0695261884',
     'EP0939931413','EF0273763788','GP0035242171','GP0994999656',
     'SL0237577932','EH0109845675') OR PLAYER_RANK<=10
ORDER BY PLAYERNAME,PLAYER_RANK
```

Total Cost: 5105.9

Query Degree: 1

```
Rows
RETURN
( 1)
Cost
I/O
|
818.703
FILTER
( 2)
5105.9
803.1
|
2046.76
TBSCAN
( 3)
5105.44
803.1
|
2046.76
SORT
( 4)
5105.34
803.1
|
2046.76
UNION
( 5)
5102
803.1
/-----------+-----------\
22 2024.76
NLJOIN TBSCAN
( 6) ( 10)
1650.19 3451.58
66 737.1
/------+------\ |
22 1 2024.76
TBSCAN FETCH SORT
( 7) ( 8) ( 11)
0.000141703 75.0168 3451.48
0 3 737.1
| /---+--\ |
22 1 95556 2024.76
TABFNC: SYSIBM IXSCAN TABLE: DB2ADMIN FETCH
GENROW ( 9) CT_PLAYER ( 12)
50.014 3450.64
2 737.1
| /---+---\
95556 2582.59 95556
INDEX: DB2ADMIN RIDSCN TABLE: DB2ADMIN
CT_PLAYER_PK ( 13) CT_PLAYER
664.375
26.5135
|
2582.59
SORT
( 14)
664.375
26.5135
|
2582.59
IXSCAN
( 15)
663.752
26.5135
|
95556
INDEX: DB2ADMIN
CT_PLAYER_PK
```

Create Table DDL for MDC Table

```
CREATE TABLE "DB2ADMIN"."CT_PLAYER"  (
"TOURNAMENT_ID" INTEGER NOT NULL ,
"ACC_NUM" CHAR(12) NOT NULL ,
"STAKE" DECIMAL(11,2) NOT NULL ,
"PLAY_COUNT" INTEGER NOT NULL ,
"FINAL_POSITION" INTEGER ,
"REAL_PRIZE_PAID" DECIMAL(11,2) ,
"PROMO_PRIZE_PAID" DECIMAL(11,2) ,
"LOCK" TIMESTAMP NOT NULL WITH DEFAULT CURRENT TIMESTAMP ,
"REBUY_COUNT" INTEGER NOT NULL WITH DEFAULT 0 ,
"REBUY" CHAR(1) NOT NULL WITH DEFAULT 'F' ,
"TOKEN_ID" VARCHAR(25) ,
"BUYIN_TYPE" CHAR(1) NOT NULL WITH DEFAULT 'R' ,
"GAME_ID" INTEGER NOT NULL WITH DEFAULT 0 ,
"NUM_QUALIFY" INTEGER WITH DEFAULT -1 ,
"PLAYERNAME" VARCHAR(20) ,
"UPDATE_TS" TIMESTAMP NOT NULL WITH DEFAULT  )
IN "TSD_SIN" INDEX IN "TSI_SIN"
ORGANIZE BY (
( "TOURNAMENT_ID" ) )
```

Cost of Query After MDC Table Created

Total Cost: 765.726 **85%**
Query Degree: **Improvement**

```
Rows
RETURN
( 1)
Cost
I/O
|
800.583
FILTER
( 2)
765.726
192
|
2001.46
TBSCAN
( 3)
765.274
192
|
2001.46
SORT
( 4)
765.179
192
|
2001.46
UNION
( 5)
761.921
192
```

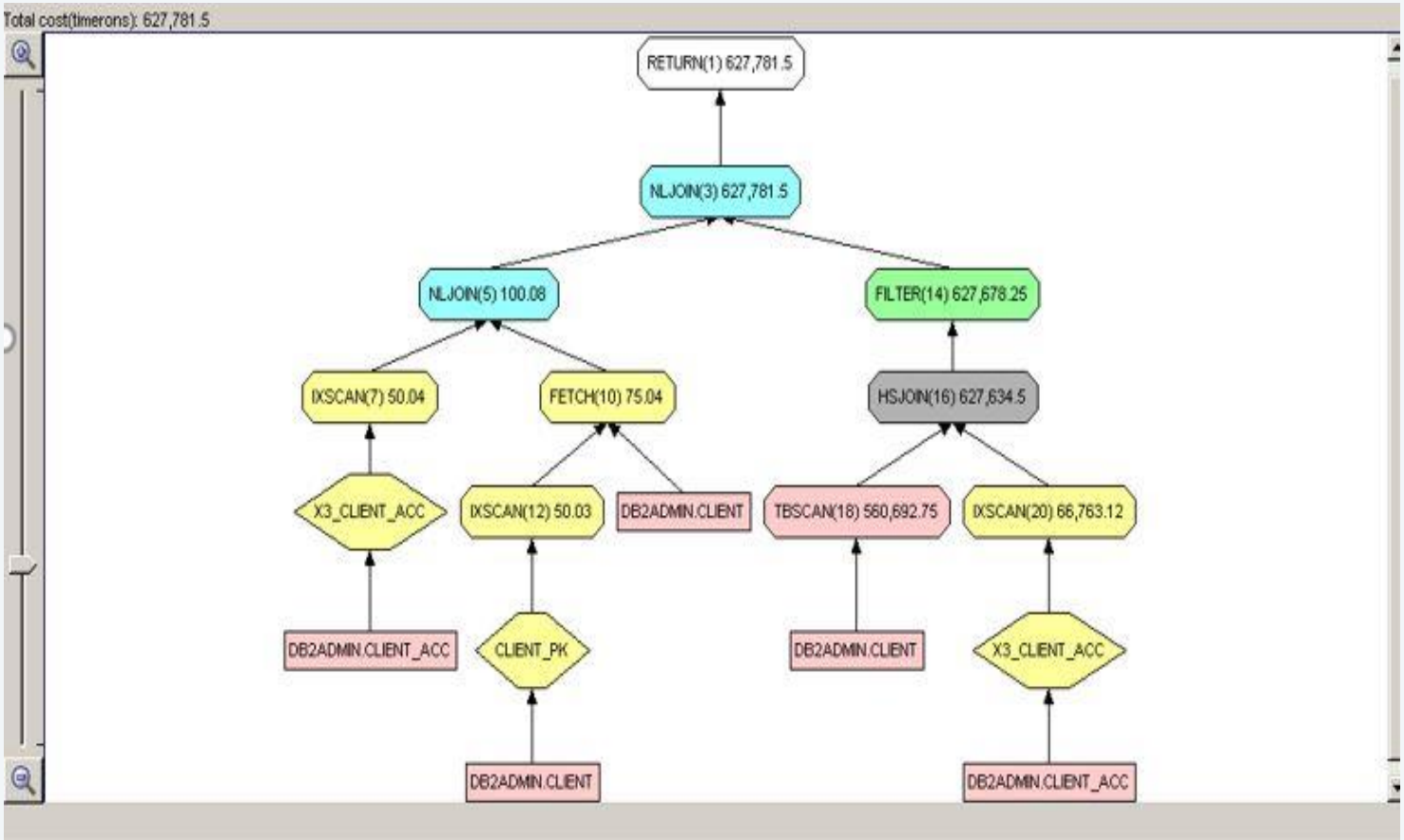
```
                                        /-------------+-------------\
                                      22                        1979.46
                                   MSJOIN                      TBSCAN
                                     ( 6)                       ( 15)
                                 381.049                      380.646
                                      96                           96
                                /--------+-------\                   |
                     2324.66          0.00946376            1979.46
                      TBSCAN            FILTER               SORT
                       ( 7)             ( 11)                ( 16)
                     380.844          0.0078653             380.552
                          96                0                   96
                           |                |                    |
                     2324.66             22                 1979.46
                      SORT             TBSCAN               FETCH
                       ( 8)             ( 12)                ( 17)
                     380.844          0.0078653             379.727
                          96                0                   96
                           |                |                /---+---\
                 2324.66              22          2.45714     81363
FETCH              SORT            IXSCAN     TABLE: DB2ADMIN
  ( 9)              ( 13)           ( 18)      MDC_CT_PLAYER
                   380.141          0.00653574          2.95247
                                       96                0          0
                                   /---+---\             |          |
                  2.45714          81363          22          81363
IXSCAN     TABLE: DB2ADMIN        TBSCAN       INDEX: SYSIBM
 ( 10)      MDC_CT_PLAYER         ( 14)       SQL0612201328275
                                 2.95247                0.000141703
                                       0                          0
                                       |                          |
                                   81363                         22
                       INDEX: SYSIBM                     TABFNC: SYSIBM
                        SQL0612201328275                      GENROW
```

Block Index Used!
List prefetch eliminated!

More Suboptimal SQL

Total execution time (sec.ms) = 9.662979
Total user cpu time (sec.ms) = 9.328125
Total system cpu time (sec.ms) = 0.187500
Statement text = select client_acc.acc_num from client_acc,client where substr(client_acc.acc_num,1,2) != ? and substr(client_acc.acc_num,1,2) in ('SD','SF') and client_acc.client_id = client.client_id and UCASE(rtrim(client.email)) = (select UCASE(rtrim(client.email)) from client_acc,client where client_acc.client_id=client.client_id and client_acc.acc_num = ?)

Suboptimal SQL – Before Rewrite

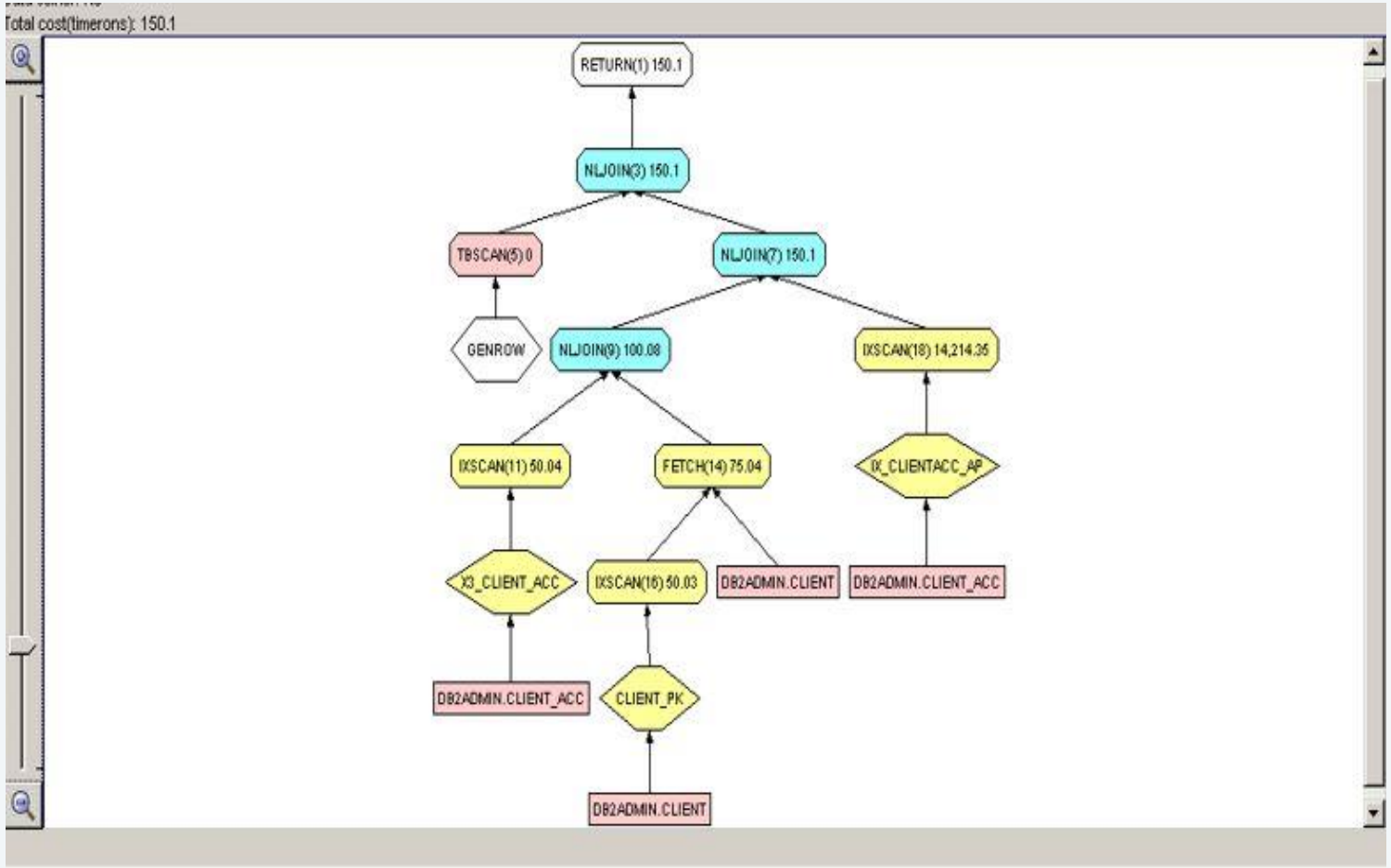


Suboptimal SQL Solution -- Encourage EARLY-OUT Sub-query

```
SELECT client_acc.acc_num
FROM   client_acc,
       client
WHERE  Substr(client_acc.acc_num, 1, 2) != ?
       AND Substr(client_acc.acc_num, 1, 2) IN ( 'SD', 'SF' )
       AND client_acc.client_id = client.client_id
       AND ( client.client_id, Ucase(Rtrim(client.email)) ) IN
          (SELECT y.client_id,
          Ucase(Rtrim(client.email))
          FROM   client_acc x,
          client y
          WHERE  client_acc.client_id = client.client_id
          AND client_acc.acc_num = ?);
```

Use of IN causes semi-join to occur and query to break out of inner loop as soon a match is found

Suboptimal SQL -- After Rewrite



DB2 11.1 SQL Improvements

- More done below the BLU line in BLU Acceleration
- Inline Optimization Hints
- Functions for regular expressions
- Insert and Update improvements for large data sets for columnar tables
- Deferred memory commit*
- Encoding dictionary support for insert and update statements for columnar tables

DB2 11.5 SQL Improvements

- Insert and Update improvements for large data sets for columnar tables
- Deferred memory commit*
- Encoding dictionary support for insert and update statements for columnar tables

Summary

- Predicate best practices discussed
- Predicate examples provided
- Problem SQL presented and various solutions provided
- Analysis of problem SQL presented
- Various solutions identified
- Importance of identifying, analyzing and tuning sub-optimal SQL highlighted
- Tips, techniques and solutions were provided

Please fill out your session evaluation before leaving!

Phil Gunning
Gunning Technology Solutions, LLC
pgunning@gts1consulting.com

Session code: F12

IDUG
Leading the Db2 User Community since 1988

Extra Slides

SQL to Identify Indexes with Jump Scans

```
SELECT varchar(tabname, 33) as tabname, iid, index_scans,
index_jump_scans from table(mon_get_index( null, null, -1)) where
index_jump_scans > 1 order by index_jump_scans desc fetch first 7
rows only
```

List of Tables with Indexes Using Jump Scan

TABNAME	IID	INDEX_SCANS	INDEX_JUMP_SCANS
---	---	---	---
BROADCAST_MESSAGE_SEGMENTS	1	6598481	6598449
REWARD_BRAND	1	244353	244259
COMPONENT_GROUP_MERCHANT	2	40773	39108
GSPFE_GAME_GAME_PROPERTY	1	2766274	34917
WEB_TRACKING	1	20722	18301
ACC_TRACE	1	31024	1604
ACC_TRACE	8	177544	470

Describe the Indexes on the Broadcast_Message_Segments Table

Index name	Un rul	Nun colu	Index type	Index parti	Null keys	Index ID	Data type	Has	Max \ Lengt	Xml patte	BUSINESS_ WITHOUT	Column names		
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
GTS_X1_BMS	D	3	RELATIC	-	Y	1	-	-	-	-	NO	+MSG_ID+CASINO_ID+SEGMENT		

SQL to Find the SQL being Executed and Using Jump Scan

```
select executable_id
from  TABLE(MON_GET_PKG_CACHE_STMT ( 'D', NULL, NULL, -2))
where upper(stmt_text) like '%BROADCAST_MESSAGE_SEGMENTS%'
  and stmt_text not like 'create or replace%'
  and ( select INDEX_JUMP_SCANS from table(mon_get_index( null,
null, -1)) where upper(TABNAME) =
'BROADCAST_MESSAGE_SEGMENTS' and iid = 1 ) >= 6542645
```

Explain the SQL using the Executable ID

db2 " call explain_from_section (
x'0100000000000000372C0D0000000000000000000200201908270218
44388000' , 'M', null, 0 , 'ADMINISTRATOR', ? , ? , ?, ?, ?)"

Use db2exfmt to Format the Explain

Connecting to the Database.

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.A

FORMATTED ON DB: TQGAMPRD

SOURCE_NAME: SYSSH100

SOURCE_SCHEMA: NULLID

SQL Using Jump Scan (from db2exfmt)

Original Statement:

```
SELECT   bm.msg_id,  create_date,  admin_id,  bms.casino_id,active_start,  active_end,
msg_type, CAST(msg_content AS VARCHAR(1024)
      FOR BIT DATA) AS msg_content, bms.segment, bm.reoccurency,  dm.delivery_count
FROM
  broadcast_message bm   join broadcast_message_segments bms
  on bm.msg_id = bms.msg_id          LEFT JOIN delivered_message dm
  ON bm.msg_id = dm.msg_id AND  dm.acc_num = ?
WHERE
  current timestamp between active_start and active_end       AND coalesce(cancel, 'N') != 'Y'
            AND bm.reoccurency > coalesce(dm.delivery_count, -1)          AND bms.casino_id =
? AND  bms.segment in (?, ?) ORDER BY  active_start WITH UR
```

```
SELECT
  bm.msg_id,  create_date,  admin_id,  bms.casino_id,   active_start,  active_end,
  msg_type, CAST(msg_content AS VARCHAR(1024)
       FOR BIT DATA) AS msg_content, bms.segment,bm.reoccurency, dm.delivery_count
FROM
  broadcast_message bm              join broadcast_message_segments bms
  on bm.msg_id = bms.msg_id       LEFT JOIN delivered_message dm
  ON bm.msg_id = dm.msg_id AND
  dm.acc_num = ?
WHERE
  current timestamp between active_start and active_end              AND coalesce(cancel, 'N') != 'Y'    AND
bm.reoccurency > coalesce(dm.delivery_count, -1)        AND bms.casino_id = ? AND
  bms.segment in (?, ?)
ORDER BY
  active_start
```

Jump Scan used in Step 20 of db2exfmt

```
20) IXSCAN: (Index Scan)
                    Cumulative Total Cost:                  45.6775
                    Cumulative First Row Cost:    45.0315
                    Estimated Bufferpool Buffers:           3299
                    Arguments:
                    ---------
                    JUMPSCAN: (Jump Scan Plan)
                              TRUE
                    Gap Info:       Status
                    ---------       ------
                    Index Column 1:     No Gap
                    Index Column 2:     Gap
                    Index Column 3:     No Gap
                    Input Streams:
                    -------------
                              13) From Object DB2ADMIN.GTS_X1_BMS
                                        Estimated number of rows:   557581
                                        Column Names:
                                        ------------
```