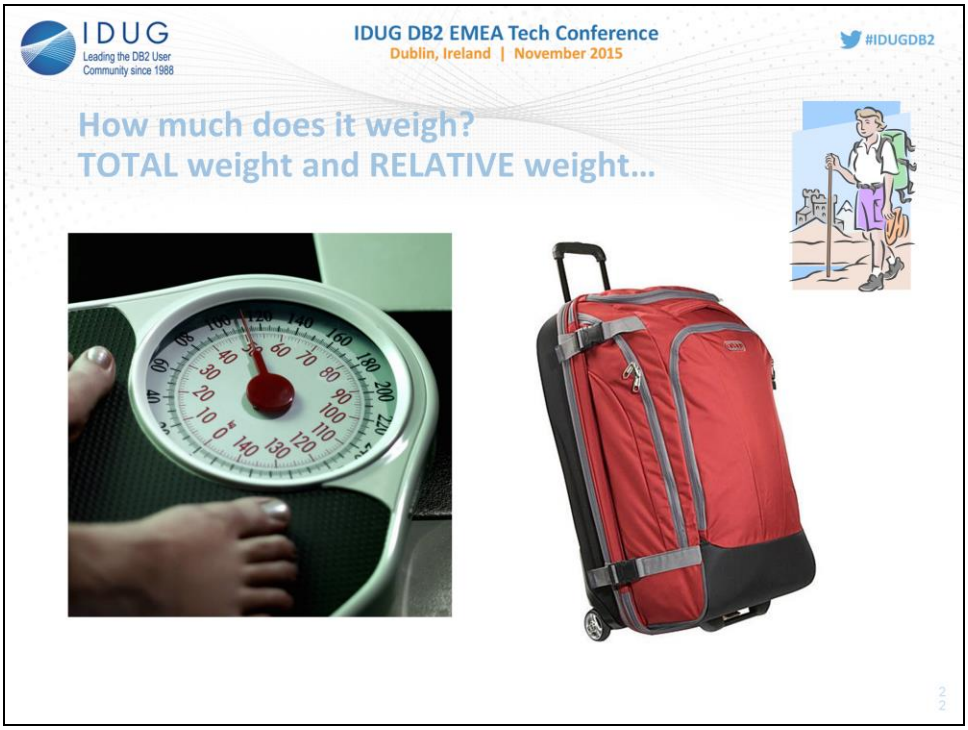

IDUG
Leading the DB2 User Community since 1988
IDUG DB2 EMEA Tech Conference
Dublin, Ireland | November 2015
#IDUGDB2
More Sage Advice: Invaluable DB2 LUW Performance Insights from Around the Globe
Scott Richard Hayes, DBI Software
Session Code: D10
Wednesday, 18 November, 8:30-9:30
Platform: DB2 for Linux, UNIX, and Windows



The scale shows the total weight of a person with ugly toes.

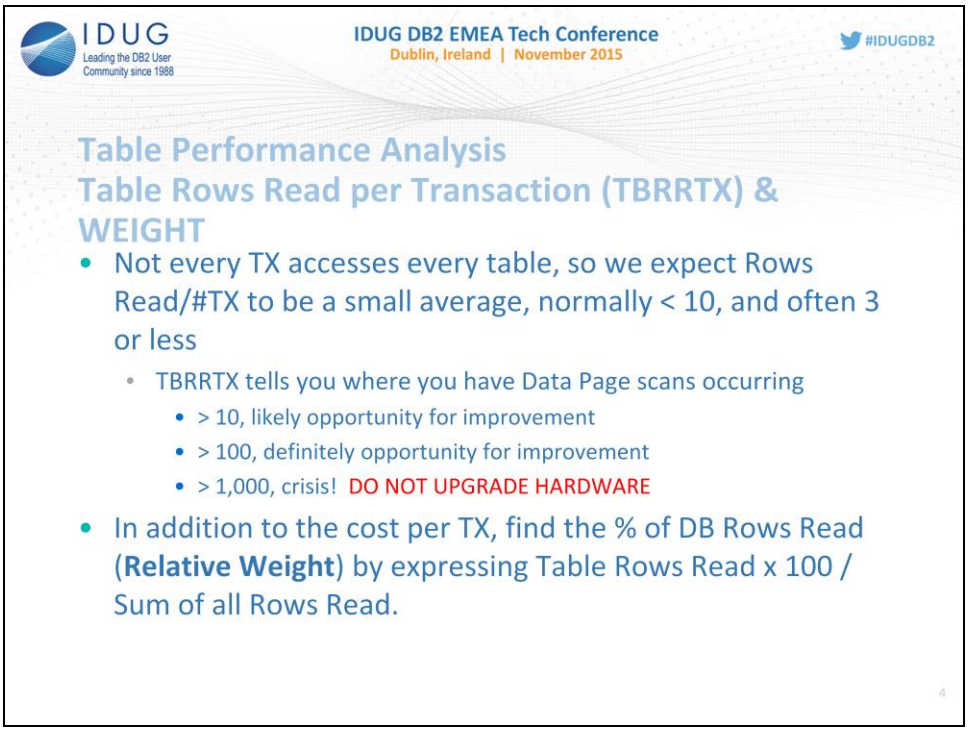
If you were going hiking, how much might your back pack weigh?

What's in your backpack?

Does it weigh too much?

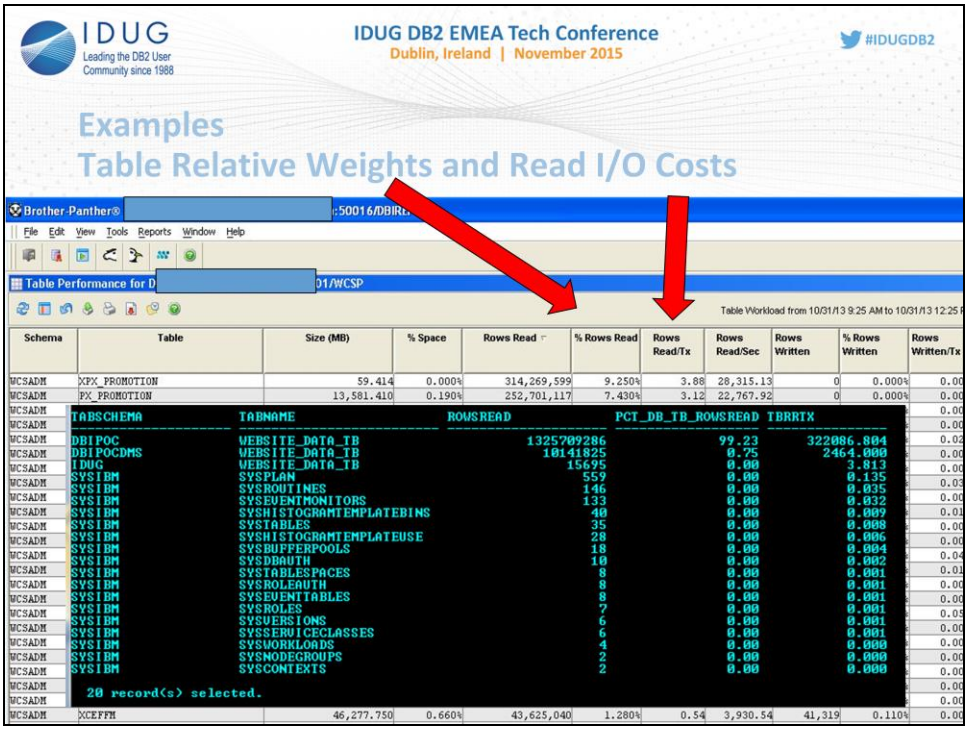
If you want the back pack to weigh less, and you have limited time, what might you remove?

When your database is carrying too much weight, forward motion stops! Or sputters forward at a snail's pace… and your phone invariably rings! "The database is slow!" Now you need to lessen the weight burden, and fast!!



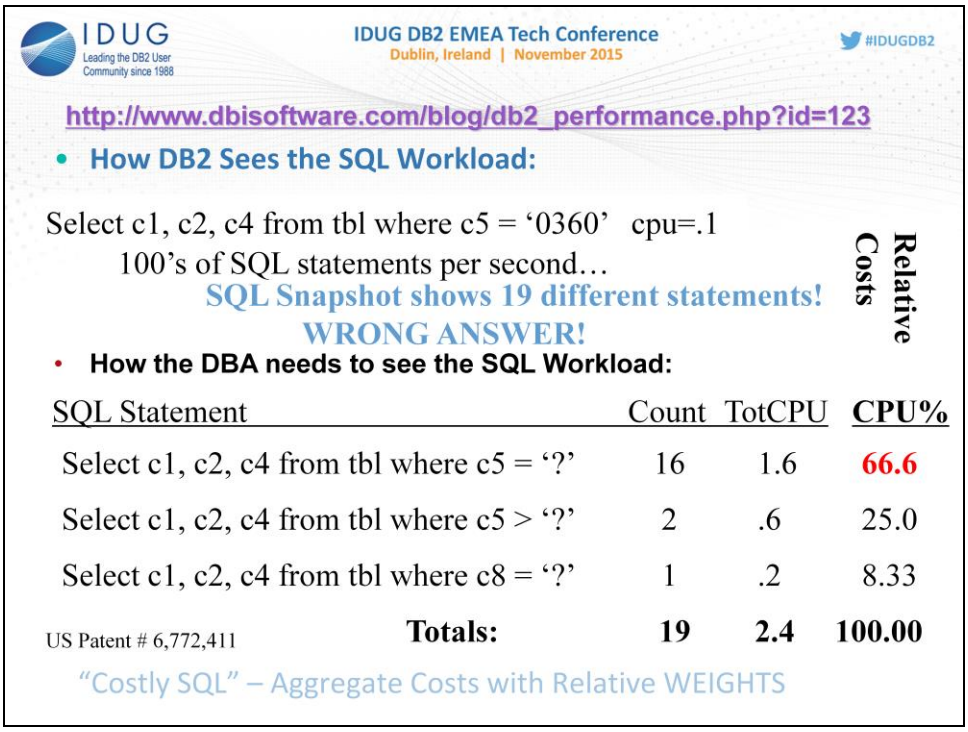

```
-- Author: Scott.Hayes@DBIsoftware.com
-- Version: 3.0
-- Last Updated: 2014-03-06
-- Copyright 2014 DBI.  All Rights Reserved.
-- Licensed for use by IDUG 2014 Attendees and DBI Authorized Customers Only
--
select substr(a.tabschema,1,20) as TABSCHEMA,
      substr(a.tabname,1,25) as TABNAME,
      a.rows_read as RowsRead,
      CAST((((A.ROWS_READ) * 100.0)
       / (Select (SUM(Z.ROWS_READ) + 1.0)
            FROM SYSIBMADM.SNAPTAB Z
           WHERE A.DBPARTITIONNUM = Z.DBPARTITIONNUM
            )) AS DECIMAL(5,2)) AS PCT_DB_TB_ROWSREAD,
      CAST( (a.rows_read / (b.commit_sql_stmts + b.rollback_sql_stmts + 1.0))
       AS DECIMAL(13,3)) as TBRRTX
```

```
  from SYSIBMADM.snaptab a,
       SYSIBMADM.snapdb b
where a.dbpartitionnum = b.dbpartitionnum
order by a.rows_read desc fetch first 20 rows only;
```

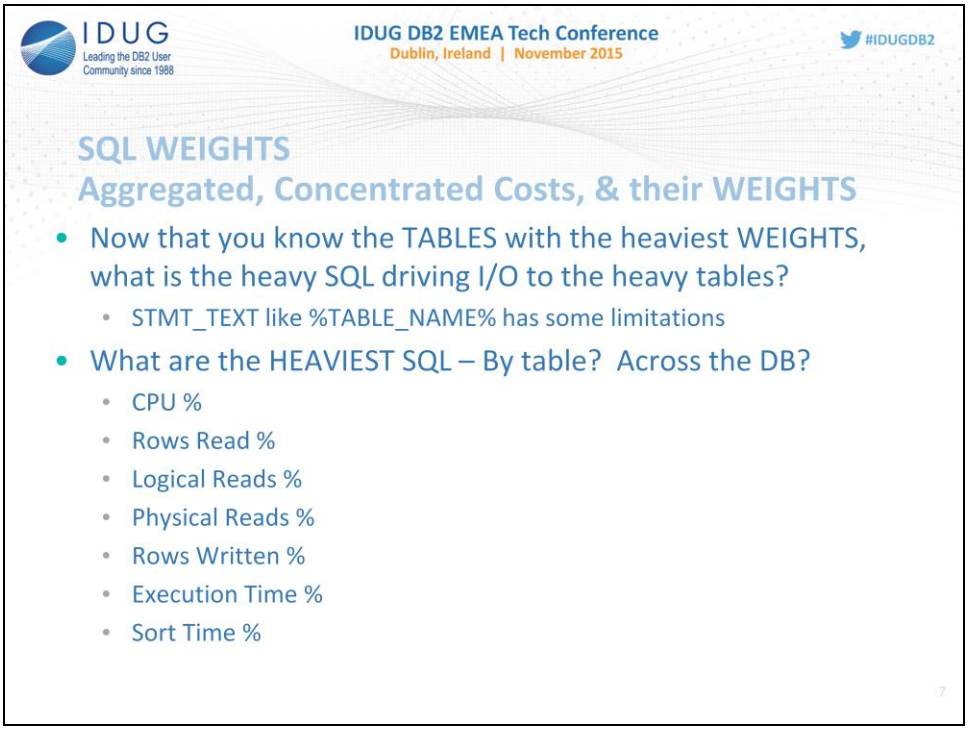



```
-- Author: Scott.Hayes@DBIsoftware.com
-- Version: 3.0
-- Last Updated: 2014-03-06
-- Copyright 2014 DBI.  All Rights Reserved.
-- Licensed for use by IDUG 2014 Attendees and DBI Authorized Customers Only
--
select substr(a.tabschema,1,20) as TABSCHEMA,
       substr(a.tabname,1,25) as TABNAME,
       a.rows_read as RowsRead,
       CAST((((A.ROWS_READ) * 100.0)
        / (Select (SUM(Z.ROWS_READ) + 1.0)
             FROM SYSIBMADM.SNAPTAB Z
            WHERE A.DBPARTITIONNUM = Z.DBPARTITIONNUM
             )) AS DECIMAL(5,2)) AS PCT_DB_TB_ROWSREAD,
       CAST( (a.rows_read / (b.commit_sql_stmts + b.rollback_sql_stmts + 1.0))
        AS DECIMAL(13,3)) as TBRRTX
```

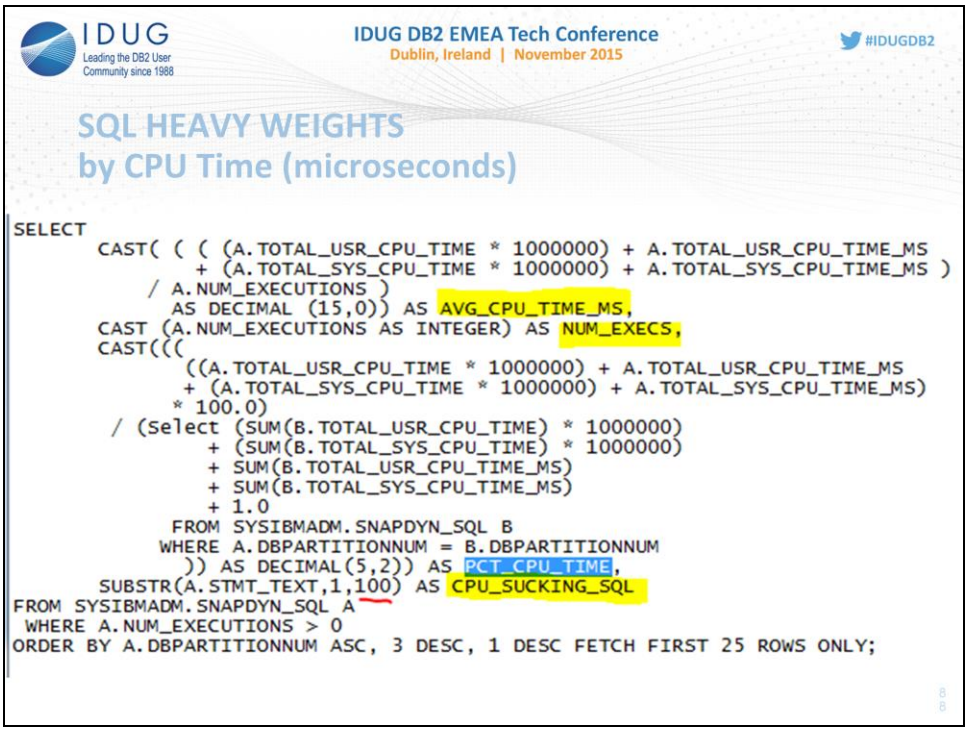
```
  from SYSIBMADM.snaptab a,
       SYSIBMADM.snapdb b
where a.dbpartitionnum = b.dbpartitionnum
order by a.rows_read desc fetch first 20 rows only;
```



SQL Cost aggregation is imperative to successfully understanding statement workload costs. In the absence of aggregating statement costs, you are merely hunting elephants. When costs are aggregated, your analysis may likely provide shocking and invaluable insights into performance. More help on this topic is available at http://www.dbisoftware.com/help/index.php



For relative weights to be invaluable, you have to know the total aggregate costs of execution independent of any literal values that might be present in the SQL. For static SQL, and dynamic SQL with parameter markers, or if you have the STMT_CONC = LITERALS database configuration parameter set, cost aggregation is done correctly by DB2. But, the majority of SQL in the world is dynamic with literals, and only a very small percentage (about 10% in a recent DB2Night Show audience survey) of DB2 customers use the statement concentrator due to potential adverse side effects.



-- SQL Analysis - SQL with Execution Times: SQLCPUTIME.SQL

-- Results best viewed with command Window 200 characters wide

-- Author: Scott.Hayes@DBIsoftware.com

-- Version: 1.0

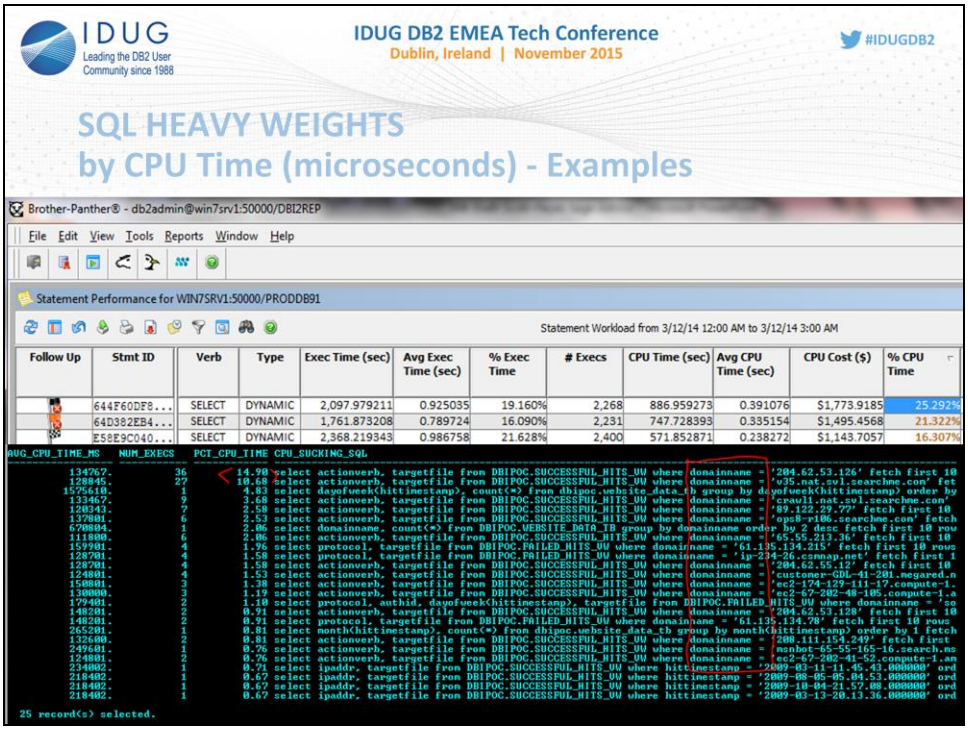
-- Last Updated: 2014-03-12

-- Copyright 2014 DBI. All Rights Reserved.

-- Licensed for use by paid IDUG 2014 Attendees and Authorized DBI Customers Only

```
SELECT
      CAST( ( ( (A.TOTAL_USR_CPU_TIME * 1000000) +
A.TOTAL_USR_CPU_TIME_MS
            + (A.TOTAL_SYS_CPU_TIME * 1000000) +
A.TOTAL_SYS_CPU_TIME_MS )
         / A.NUM_EXECUTIONS )
          AS DECIMAL (15,0)) AS AVG_CPU_TIME_MS,
      CAST (A.NUM_EXECUTIONS AS INTEGER) AS NUM_EXECS,
      CAST(((
          ((A.TOTAL_USR_CPU_TIME * 1000000) +
A.TOTAL_USR_CPU_TIME_MS
```

```
          + (A.TOTAL_SYS_CPU_TIME * 1000000) + A.TOTAL_SYS_CPU_TIME_MS)
         * 100.0)
      / (Select (SUM(B.TOTAL_USR_CPU_TIME) * 1000000) +
(SUM(B.TOTAL_SYS_CPU_TIME) * 1000000)
               + SUM(B.TOTAL_USR_CPU_TIME_MS) +
SUM(B.TOTAL_SYS_CPU_TIME_MS) + 1.0
FROM SYSIBMADM.SNAPDYN_SQL B
         WHERE A.DBPARTITIONNUM = B.DBPARTITIONNUM
          )) AS DECIMAL(5,2)) AS PCT_CPU_TIME,
     SUBSTR(A.STMT_TEXT,1,100) AS CPU_SUCKING_SQL
FROM SYSIBMADM.SNAPDYN_SQL A
 WHERE A.NUM_EXECUTIONS > 0
ORDER BY A.DBPARTITIONNUM ASC, 3 DESC, 1 DESC FETCH FIRST 25 ROWS
ONLY;
```



-- SQL Analysis - SQL with Execution Times: SQLCPUTIME.SQL

-- Results best viewed with command Window 200 characters wide

-- Author: Scott.Hayes@DBIsoftware.com

-- Version: 1.0

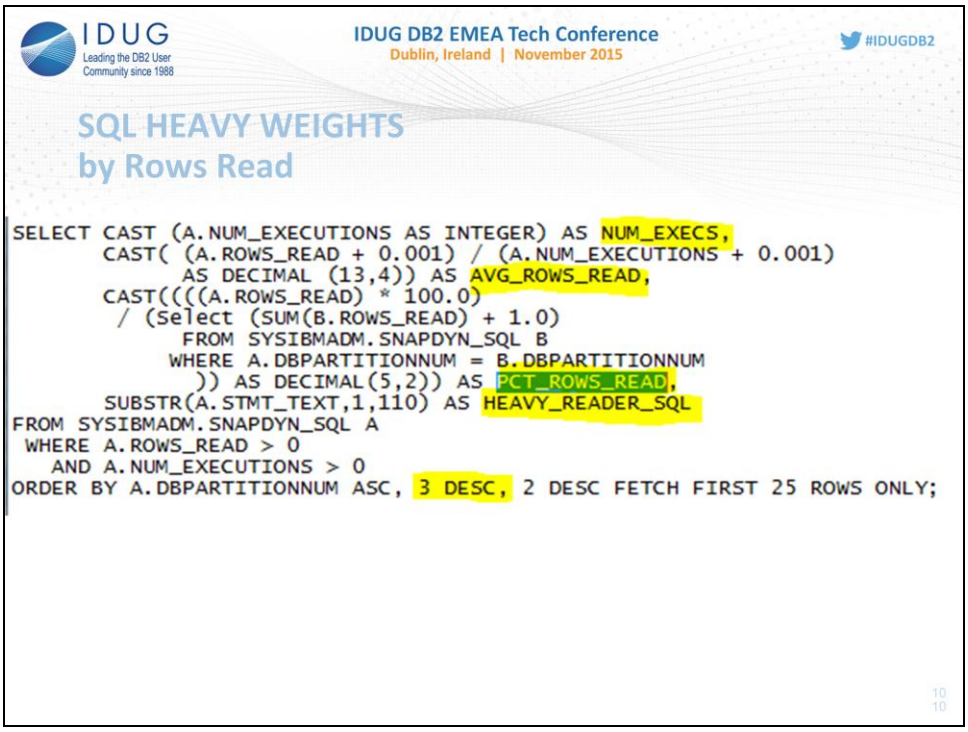
-- Last Updated: 2014-03-12

-- Copyright 2014 DBI. All Rights Reserved.

-- Licensed for use by paid IDUG 2014 Attendees and Authorized DBI Customers Only

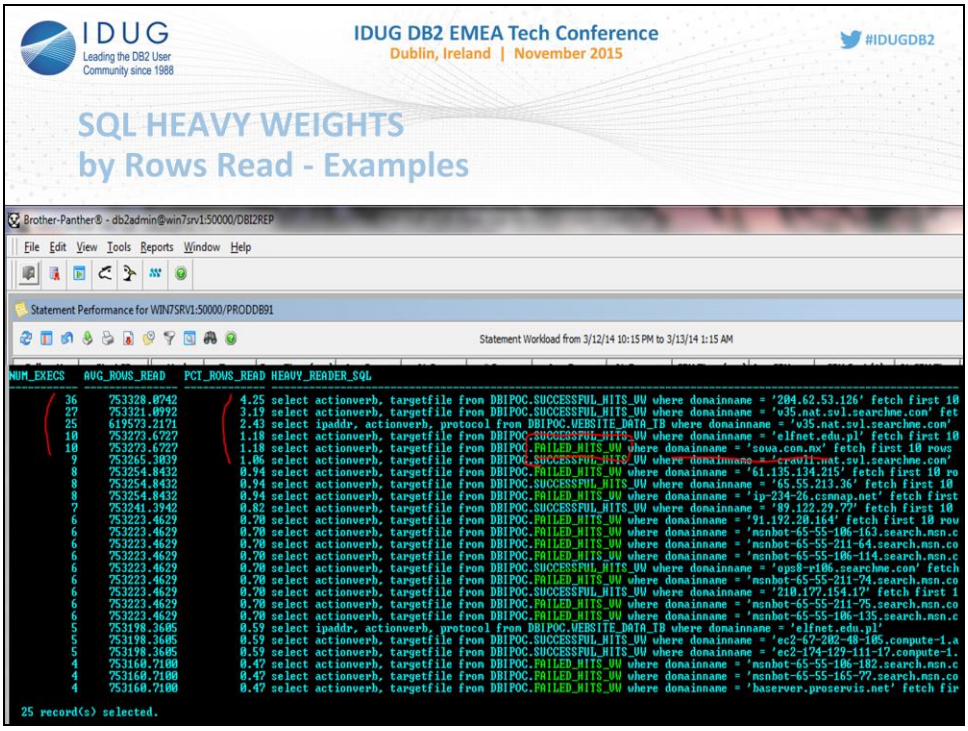
```
SELECT
      CAST( ( ( (A.TOTAL_USR_CPU_TIME * 1000000) +
A.TOTAL_USR_CPU_TIME_MS
            + (A.TOTAL_SYS_CPU_TIME * 1000000) +
A.TOTAL_SYS_CPU_TIME_MS )
         / A.NUM_EXECUTIONS )
          AS DECIMAL (15,0)) AS AVG_CPU_TIME_MS,
     CAST (A.NUM_EXECUTIONS AS INTEGER) AS NUM_EXECS,
     CAST(((
          ((A.TOTAL_USR_CPU_TIME * 1000000) +
A.TOTAL_USR_CPU_TIME_MS
```

```
          + (A.TOTAL_SYS_CPU_TIME * 1000000) + A.TOTAL_SYS_CPU_TIME_MS)
         * 100.0)
      / (Select (SUM(B.TOTAL_USR_CPU_TIME) * 1000000) +
(SUM(B.TOTAL_SYS_CPU_TIME) * 1000000)
               + SUM(B.TOTAL_USR_CPU_TIME_MS) +
SUM(B.TOTAL_SYS_CPU_TIME_MS) + 1.0
FROM SYSIBMADM.SNAPDYN_SQL B
         WHERE A.DBPARTITIONNUM = B.DBPARTITIONNUM
          )) AS DECIMAL(5,2)) AS PCT_CPU_TIME,
     SUBSTR(A.STMT_TEXT,1,100) AS CPU_SUCKING_SQL
FROM SYSIBMADM.SNAPDYN_SQL A
 WHERE A.NUM_EXECUTIONS > 0
ORDER BY A.DBPARTITIONNUM ASC, 3 DESC, 1 DESC FETCH FIRST 25 ROWS
ONLY;
```




```
-- SQL Analysis - Heavy Read I/O SQL: SQLROWSREAD.SQL
-- Results best viewed with command Window 200 characters wide
-- Author: Scott.Hayes@DBIsoftware.com
-- Version: 1.0
-- Last Updated: 2014-03-13
-- Copyright 2014 DBI.  All Rights Reserved.
-- Licensed for use by paid IDUG 2014 Attendees and Authorized DBI Customers Only
SELECT CAST (A.NUM_EXECUTIONS AS INTEGER) AS NUM_EXECS,
      CAST( (A.ROWS_READ + 0.001) / (A.NUM_EXECUTIONS + 0.001)
           AS DECIMAL (13,4)) AS AVG_ROWS_READ,
      CAST((((A.ROWS_READ) * 100.0)
       / (Select (SUM(B.ROWS_READ) + 1.0)
           FROM SYSIBMADM.SNAPDYN_SQL B
          WHERE A.DBPARTITIONNUM = B.DBPARTITIONNUM
           )) AS DECIMAL(5,2)) AS PCT_ROWS_READ,
      SUBSTR(A.STMT_TEXT,1,110) AS HEAVY_READER_SQL
FROM SYSIBMADM.SNAPDYN_SQL A
```

```
 WHERE A.ROWS_READ > 0 AND A.NUM_EXECUTIONS > 0
ORDER BY A.DBPARTITIONNUM ASC, 3 DESC, 2 DESC FETCH FIRST 25 ROWS
ONLY;
```

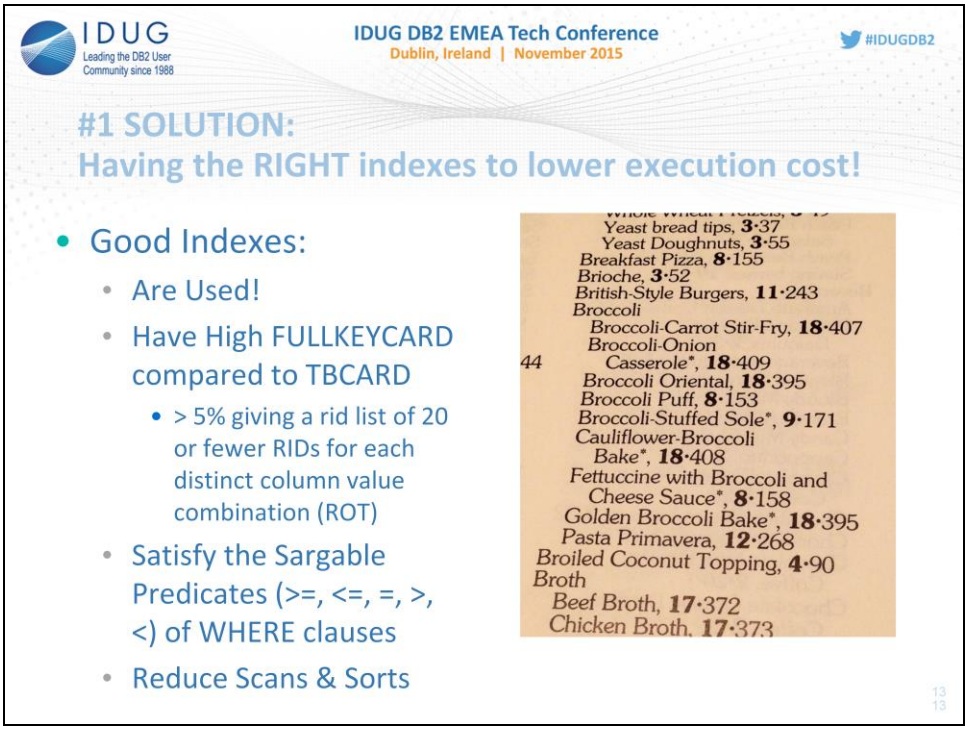



```
-- SQL Analysis - Heavy Read I/O SQL: SQLROWSREAD.SQL
-- Results best viewed with command Window 200 characters wide
-- Author: Scott.Hayes@DBIsoftware.com
-- Version: 1.0
-- Last Updated: 2014-03-13
-- Copyright 2014 DBI.  All Rights Reserved.
-- Licensed for use by paid IDUG 2014 Attendees and Authorized DBI Customers Only
SELECT CAST (A.NUM_EXECUTIONS AS INTEGER) AS NUM_EXECS,
      CAST( (A.ROWS_READ + 0.001) / (A.NUM_EXECUTIONS + 0.001)
           AS DECIMAL (13,4)) AS AVG_ROWS_READ,
      CAST((((A.ROWS_READ) * 100.0)
       / (Select (SUM(B.ROWS_READ) + 1.0)
           FROM SYSIBMADM.SNAPDYN_SQL B
          WHERE A.DBPARTITIONNUM = B.DBPARTITIONNUM
           )) AS DECIMAL(5,2)) AS PCT_ROWS_READ,
      SUBSTR(A.STMT_TEXT,1,110) AS HEAVY_READER_SQL
FROM SYSIBMADM.SNAPDYN_SQL A
```

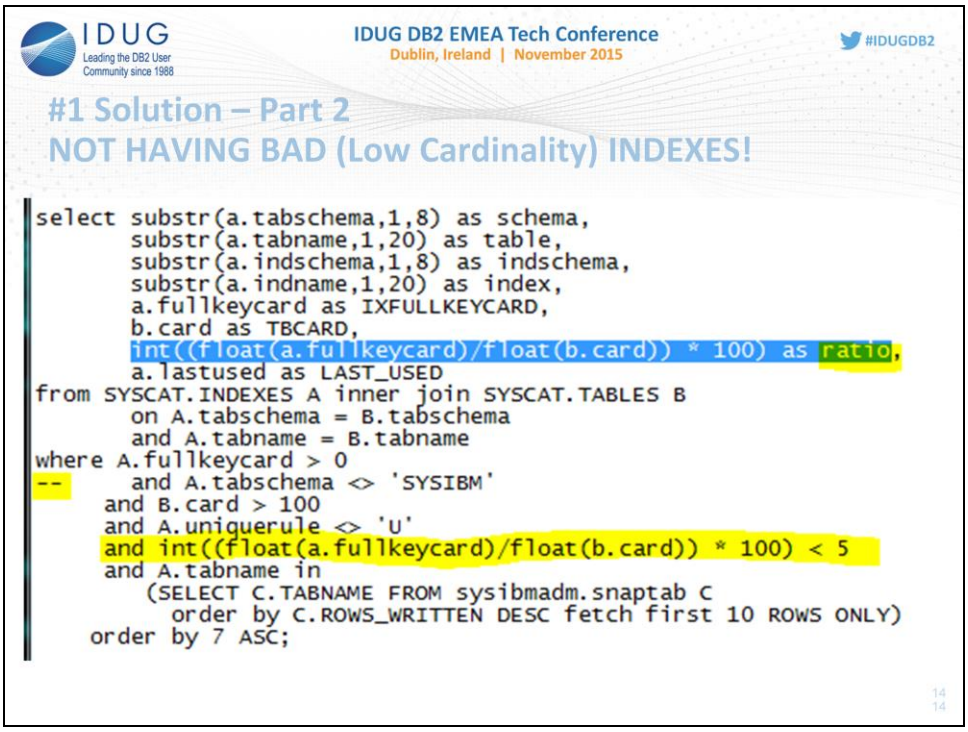
```
 WHERE A.ROWS_READ > 0 AND A.NUM_EXECUTIONS > 0
ORDER BY A.DBPARTITIONNUM ASC, 3 DESC, 2 DESC FETCH FIRST 25 ROWS
ONLY;
```


IDUG
Leading the DB2 User Community since 1988
IDUG DB2 EMEA Tech Conference
Dublin, Ireland | November 2015
#IDUGDB2
"If you call someone and get voicemail, LEAVE A MESSAGE!"
-Scott Hayes
#WISDOM

Doodle here.



Query SYSCAT.INDEXES LASTUSED column to see when an index was last used. This works reasonably well for DB2 9.7 FP5 and higher. The column is updated periodically and automatically by DB2.



-- ANALYZE Indexes for Low IXCARD on High Write Tables: IXLOWCARDV3.SQL

-- INDEXES must be HIGH QUALITY on Top 10 Write I/O Tables

-- Results best viewed with command Window ** 200 characters wide **

-- Author: Scott.Hayes@DBIsoftware.com

-- Version: 3.0

-- Last Updated: 2014-03-14

-- Licensed for use by paid IDUG Attendees and DBI Authorized Customers Only

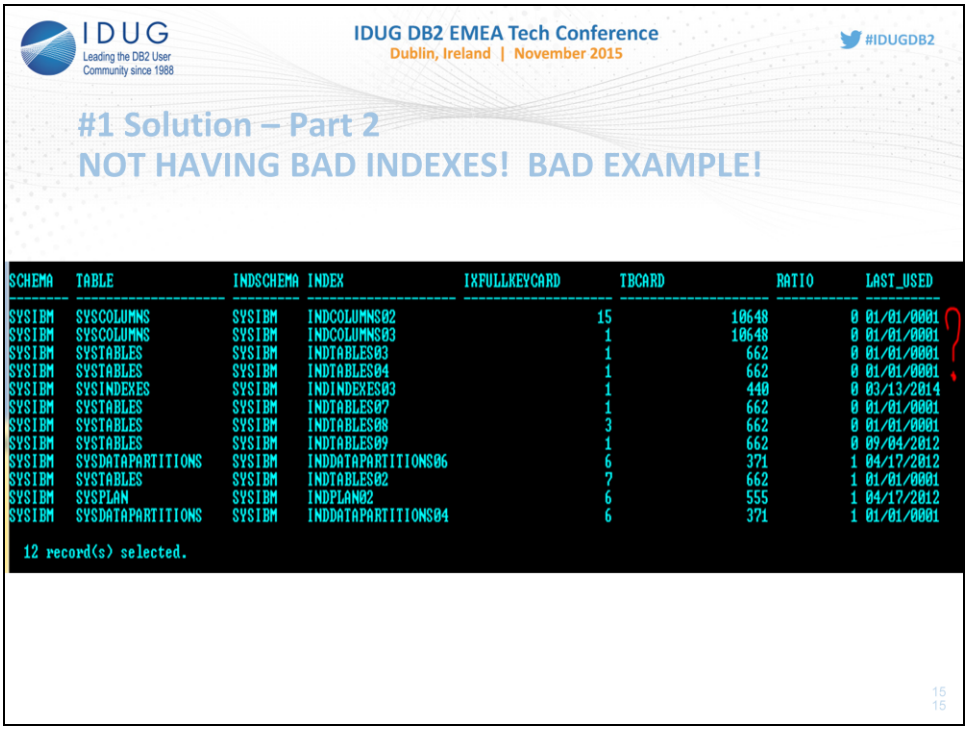
--

-- For the top 10 most highly written to tables, indentify the indexes having very low cardinality

-- compared to the table cardinality.

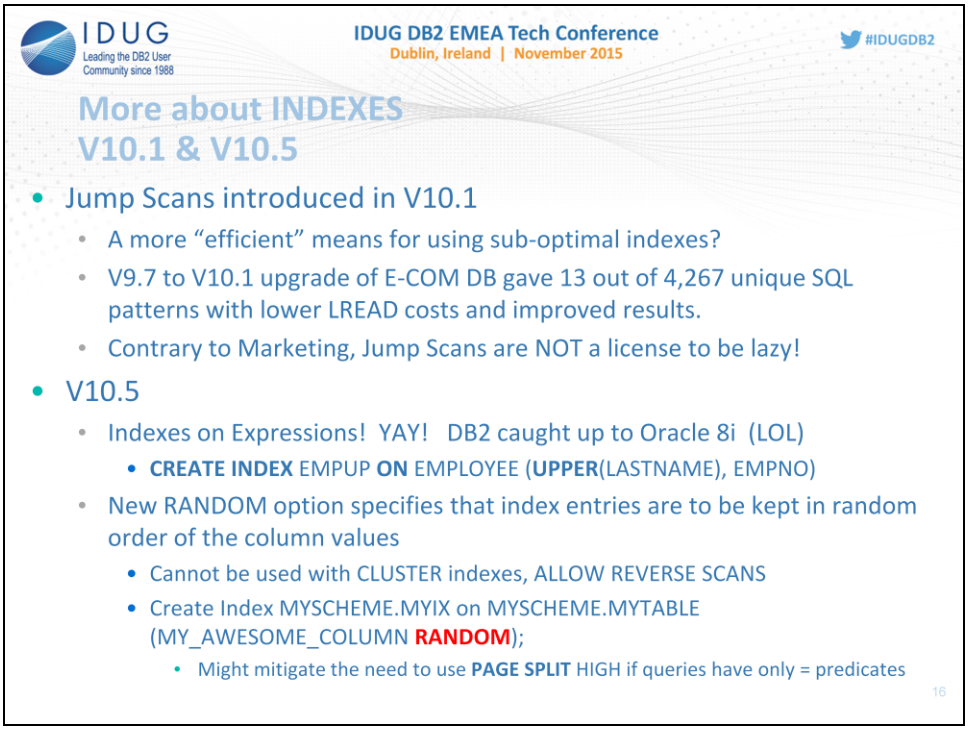
```
select substr(a.tabschema,1,8) as schema, substr(a.tabname,1,20) as table,
       substr(a.indschema,1,8) as indschema, substr(a.indname,1,20) as index,
       a.fullkeycard as IXFULLKEYCARD, b.card as TBCARD,
```

```
        int((float(a.fullkeycard)/float(b.card)) * 100) as ratio,
        a.lastused as LAST_USED
from SYSCAT.INDEXES A inner join SYSCAT.TABLES B
      on A.tabschema = B.tabschema
      and A.tabname = B.tabname
where A.fullkeycard > 0
--     and A.tabschema <> 'SYSIBM'
    and B.card > 100 and A.uniquerule <> 'U'
    and int((float(a.fullkeycard)/float(b.card)) * 100) < 5
    and A.tabname in
      (SELECT C.TABNAME FROM sysibmadm.snaptab C
       order by C.ROWS_WRITTEN DESC fetch first 10 ROWS ONLY)
   order by 7 ASC;
```




```
-- ANALYZE Indexes for Low IXCARD on High Write Tables: IXLOWCARDV3.SQL
-- INDEXES must be HIGH QUALITY on Top 10 Write I/O Tables
-- Results best viewed with command Window ** 200 characters wide **
-- Author: Scott.Hayes@DBIsoftware.com
-- Version: 3.0
-- Last Updated: 2014-03-14
-- Copyright 2014 DBI.  All Rights Reserved.
-- Licensed for use by paid IDUG Attendees and DBI Authorized Customers Only
--
-- For the top 10 most highly written to tables, indentify the indexes having very low cardinality
-- compared to the table cardinality.
select substr(a.tabschema,1,8) as schema, substr(a.tabname,1,20) as table,
       substr(a.indschema,1,8) as indschema, substr(a.indname,1,20) as index,
       a.fullkeycard as IXFULLKEYCARD, b.card as TBCARD,
```

```
        int((float(a.fullkeycard)/float(b.card)) * 100) as ratio,
        a.lastused as LAST_USED
from SYSCAT.INDEXES A inner join SYSCAT.TABLES B
      on A.tabschema = B.tabschema
      and A.tabname = B.tabname
where A.fullkeycard > 0
--     and A.tabschema <> 'SYSIBM'
    and B.card > 100 and A.uniquerule <> 'U'
    and int((float(a.fullkeycard)/float(b.card)) * 100) < 5
    and A.tabname in
      (SELECT C.TABNAME FROM sysibmadm.snaptab C
       order by C.ROWS_WRITTEN DESC fetch first 10 ROWS ONLY)
   order by 7 ASC;
```

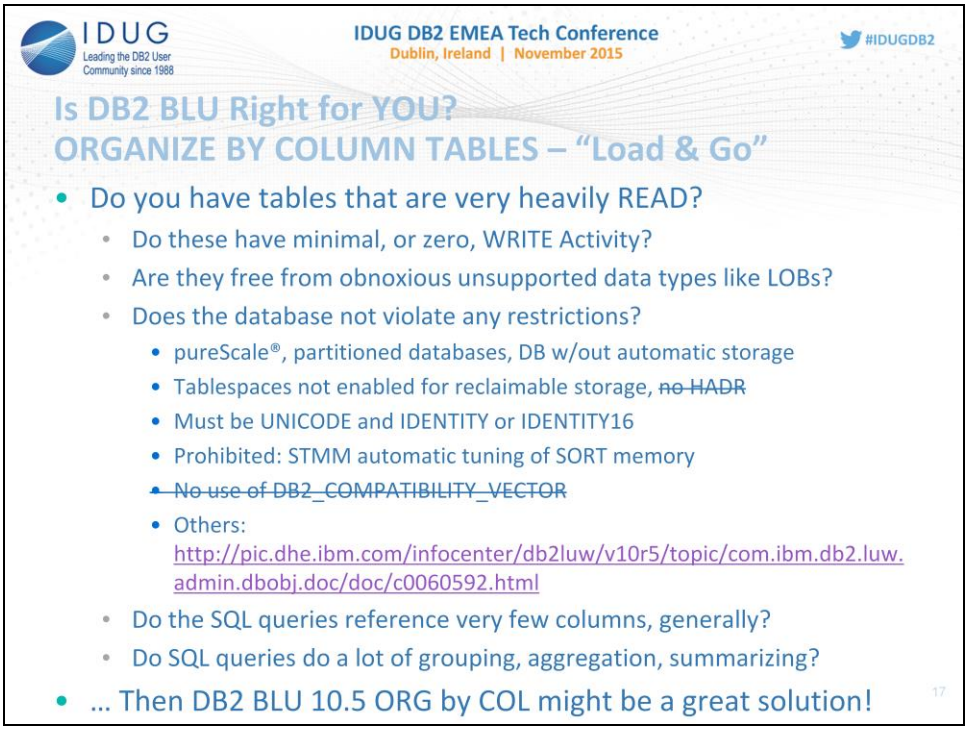


Indexes on Expressions via IBM docs:

http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0000919.html

http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/c0061101.html

The RANDOM option is intended for use by pureScale customers that experience contention/hot spots in indexes on high keys. I don't see much else practical use for it since ASC or DESC sequences can help avoid sorts and improve efficiency for accessing ranges (>, <).



Updated IBM docs:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0060592.html

Limitations:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0061528.html

The DB2Night Show™ had DB2 BLU Fest during the fall (SEPT-OCT) of 2013. Check out recorded replays of these DB2 BLU shows at http://www.DBISoftware.com/blog/db2nightshow.php

#114, DB2 BLU Early Experiences and Best Practices (IBM Canada Lab)

#115, DB2 BLU & pureScale Real Life Successes and Advice (Kent Collins, DB2 User & Consultant)

#116, Intimate Details – DB2 BLU with IBM Master Inventors (IBM Almaden Lab)

#117, Feelin' BLU – The HOTTEST IBM DB2 BLU video ever (Scott Hayes, DBI, & Randall Ibbott, QBE)

Also of potential interest:

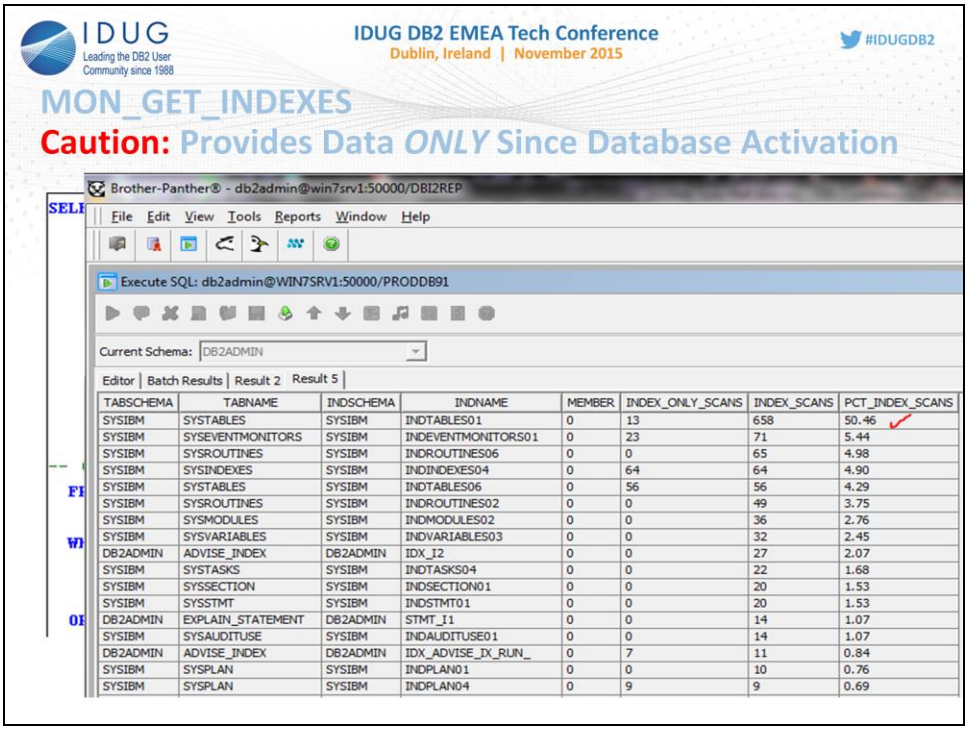
#118, 17 Laws of building very large databases! (Lee Goddard, DBI, formerly IBM)

How would you describe this piece of luggage?

Hell on wheels in a black box?

We've looked at the weight of tables plus SQL by CPU and Rows Read (Review from Sage Advice Part 1), and Part 1 also covered SQL weights by Logical Reads, Physical Reads, Rows Written, Execution Time, and Sort Time (see session D06 of IDUG NA Phoenix). Also, alternatively, watch a replay of The DB2Night Show Episode #142 to see and hear the entire presentation: http://www.dbisoftware.com/blog/db2nightshow.php?id=528

In this section of the presentation, we will contemplate some additional relative weights that merit analysis.

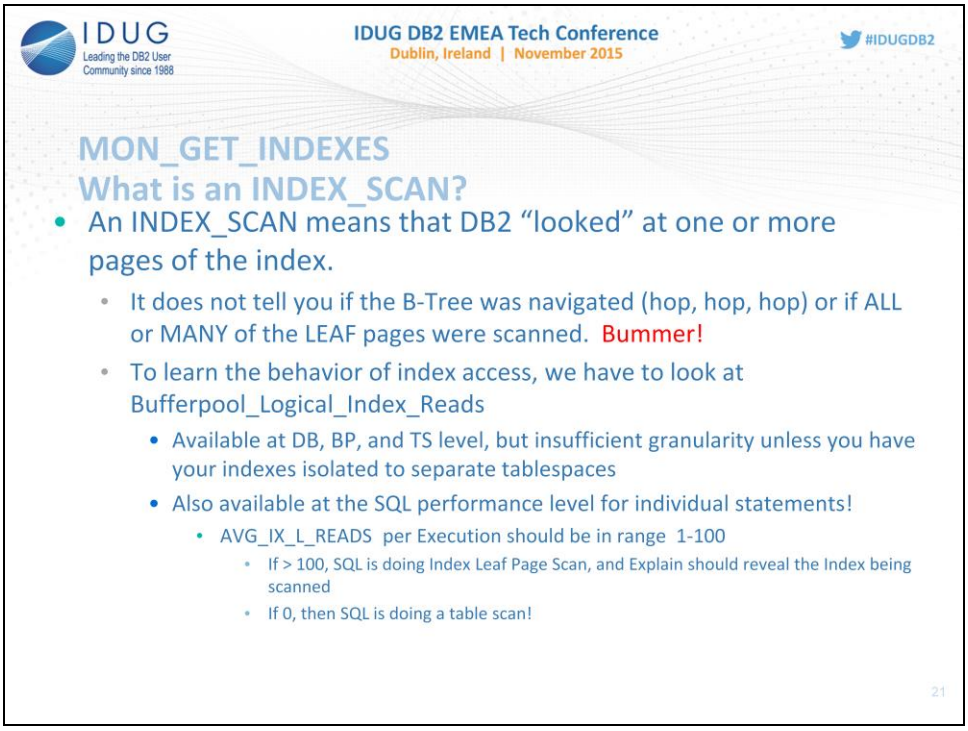


TABSCHEMA	TABNAME	INDSCHEMA	INDNAME	MEMBER	INDEX_ONLY_SCANS	INDEX_SCANS	PCT_INDEX_SCANS
SYSIBM	SYSTABLES	SYSIBM	INDTABLES01	0	13	658	50.46
SYSIBM	SYSEVENTMONITORS	SYSIBM	INDEVENTMONITORS01	0	23	71	5.44
SYSIBM	SYSROUTINES	SYSIBM	INDROUTINES06	0	0	65	4.98
SYSIBM	SYSINDEXES	SYSIBM	INDINDEXES04	0	64	64	4.90
SYSIBM	SYSTABLES	SYSIBM	INDTABLES06	0	56	56	4.29
SYSIBM	SYSROUTINES	SYSIBM	INDROUTINES02	0	0	49	3.75
SYSIBM	SYSMODULES	SYSIBM	INDMODULES02	0	0	36	2.76
SYSIBM	SYSVARIABLES	SYSIBM	INDVARIABLES03	0	0	32	2.45
DB2ADMIN	ADVISE_INDEX	DB2ADMIN	IDX_I2	0	0	27	2.07
SYSIBM	SYSTASKS	SYSIBM	INDTASKS04	0	0	22	1.68
SYSIBM	SYSSECTION	SYSIBM	INDSECTION01	0	0	20	1.53
SYSIBM	SYSSTMT	SYSIBM	INDSTMT01	0	0	20	1.53
DB2ADMIN	EXPLAIN_STATEMENT	DB2ADMIN	STMT_I1	0	0	14	1.07
SYSIBM	SYSAUDITUSE	SYSIBM	INDAUDITUSE01	0	0	14	1.07
DB2ADMIN	ADVISE_INDEX	DB2ADMIN	IDX_ADVISE_IX_RUN_	0	7	11	0.84
SYSIBM	SYSPLAN	SYSIBM	INDPLAN01	0	0	10	0.76
SYSIBM	SYSPLAN	SYSIBM	INDPLAN04	0	9	9	0.69

Query to find the heaviest weight INDEX_SCANS:

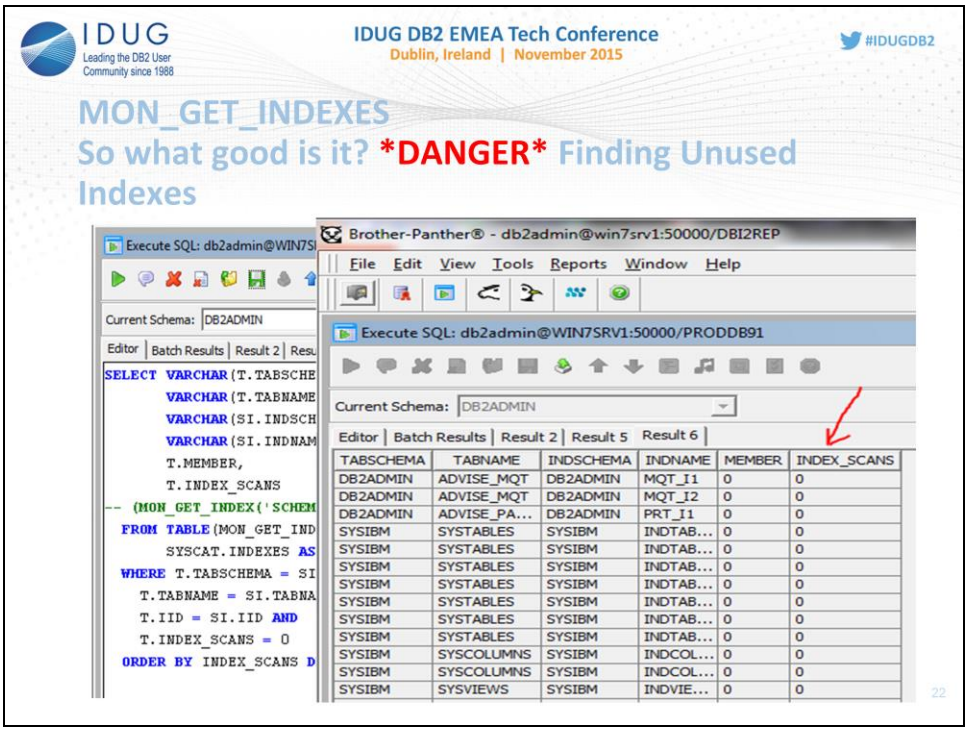
```
SELECT VARCHAR(T.TABSCHEMA, 18) AS TABSCHEMA,
      VARCHAR(T.TABNAME, 18) AS TABNAME,
      VARCHAR(SI.INDSCHEMA, 18) AS INDSCHEMA,
      VARCHAR(SI.INDNAME, 18) AS INDNAME,
      T.MEMBER,
      T.INDEX_ONLY_SCANS,
      T.INDEX_SCANS,
      CAST ((((T.INDEX_SCANS) * 100) / (SELECT
(SUM(MI.INDEX_SCANS) + 1.0)
                        FROM TABLE(MON_GET_INDEX(",", -2)) as MI ))
       AS DECIMAL(5,2)) AS PCT_INDEX_SCANS
-- (MON_GET_INDEX('SCHEMA','TABLE', MEMBER))
  FROM TABLE(MON_GET_INDEX(",", -2)) as T,
      SYSCAT.INDEXES AS SI
  WHERE T.TABSCHEMA = SI.TABSCHEMA AND
    T.TABNAME = SI.TABNAME AND
```

```
  T.IID = SI.IID
ORDER BY INDEX_SCANS DESC
```



SQL HEAVY WEIGHTS by INDEX LOGICAL READS:

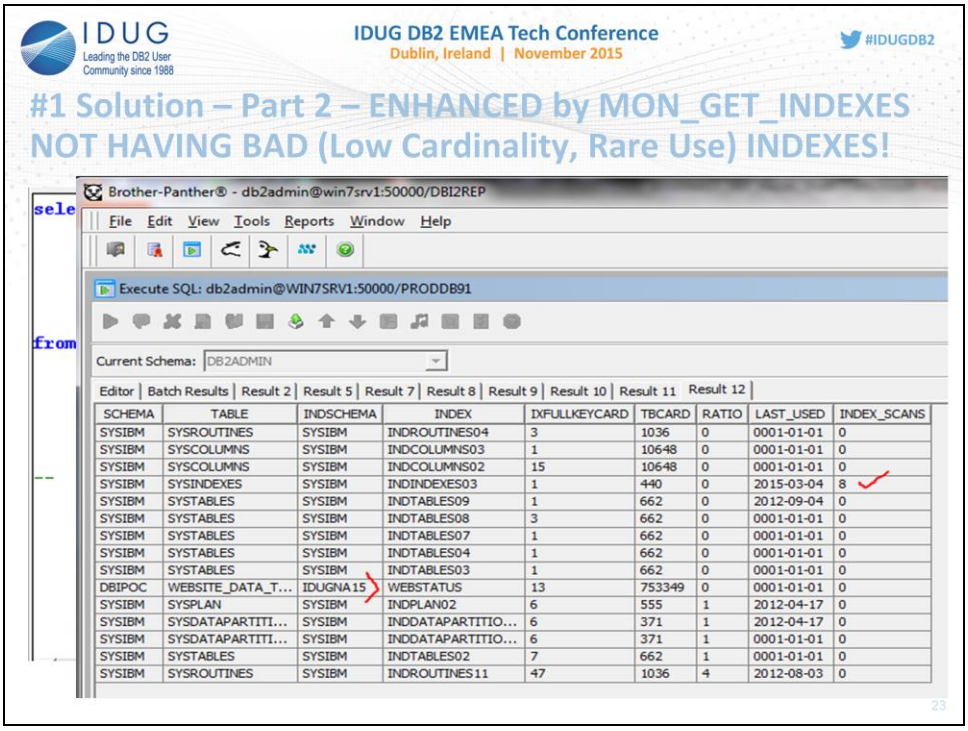
```
SELECT CAST( (A.POOL_INDEX_L_READS + 0.01) /
(A.NUM_EXECUTIONS + 0.01)
        AS DECIMAL (13,2)) AS IXLREAD_PER_EXEC,
    SUBSTR(A.STMT_TEXT,1,180) AS PROBABLE_LEAF_SCAN_SQL

  FROM SYSIBMADM.SNAPDYN_SQL A
ORDER BY A.DBPARTITIONNUM ASC, 1 DESC FETCH FIRST 25 ROWS
ONLY;
```



DANGER Remember, data is only current as of db2start!! Don't know if I'd trust this unless DB2 has been up an entire year to observe all business cycles!

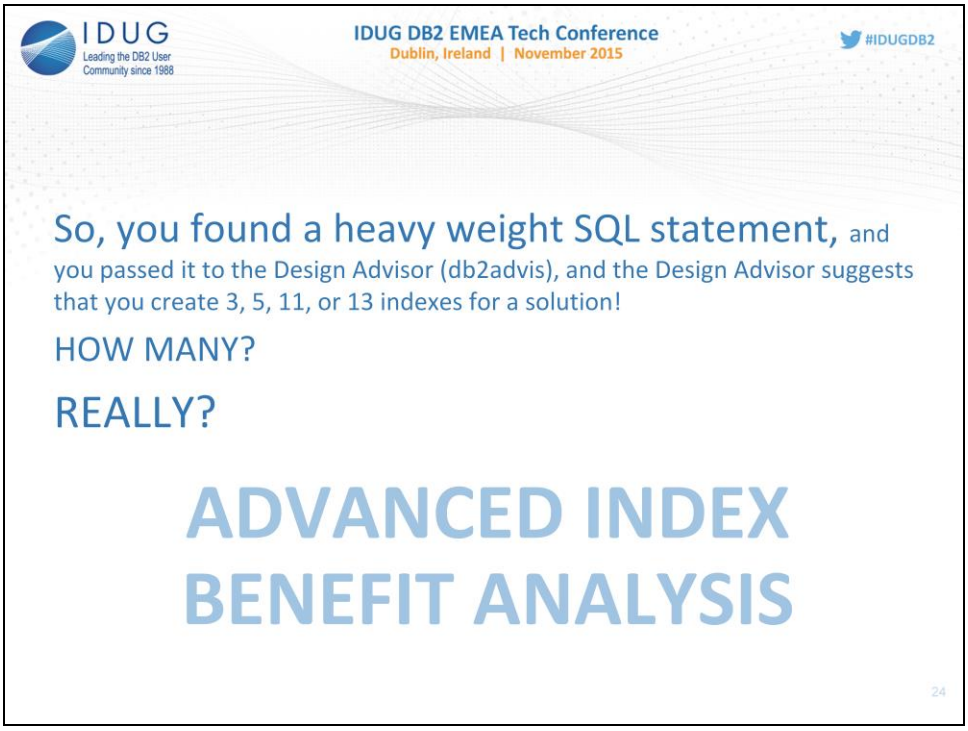
```
SELECT VARCHAR(T.TABSCHEMA, 18) AS TABSCHEMA,
       VARCHAR(T.TABNAME, 18) AS TABNAME,
       VARCHAR(SI.INDSCHEMA, 18) AS INDSCHEMA,
       VARCHAR(SI.INDNAME, 18) AS INDNAME,
       T.MEMBER,
       T.INDEX_SCANS
-- (MON_GET_INDEX('SCHEMA','TABLE', MEMBER))
  FROM TABLE(MON_GET_INDEX(",", -2)) as T,
       SYSCAT.INDEXES AS SI
  WHERE T.TABSCHEMA = SI.TABSCHEMA AND
    T.TABNAME = SI.TABNAME AND
    T.IID = SI.IID AND
    T.INDEX_SCANS = 0
```

```
ORDER BY INDEX_SCANS DESC
```




```
select substr(a.tabschema,1,8) as schema, substr(a.tabname,1,20) as table,
       substr(a.indschema,1,8) as indschema, substr(a.indname,1,20) as index,
       a.fullkeycard as IXFULLKEYCARD, b.card as TBCARD,
       int((float(a.fullkeycard)/float(b.card)) * 100) as ratio,
       a.lastused as LAST_USED, MI.INDEX_SCANS
from TABLE(MON_GET_INDEX("," , -2)) as MI, SYSCAT.INDEXES A,
SYSCAT.TABLES B
     where A.tabschema = B.tabschema and MI.tabschema = A.tabschema
       and A.tabname = B.tabname and MI.tabname = A.tabname
       and MI.IID = A.IID
     and MI.INDEX_SCANS < 100 and A.fullkeycard > 0
--     and A.tabschema <> 'SYSIBM'
     and B.card > 100 and A.uniquerule <> 'U'
     and int((float(a.fullkeycard)/float(b.card)) * 100) < 5
     and A.tabname in
       (SELECT C.TABNAME FROM sysibmadm.snaptab C
         order by C.ROWS_WRITTEN DESC fetch first 20 ROWS ONLY)
```

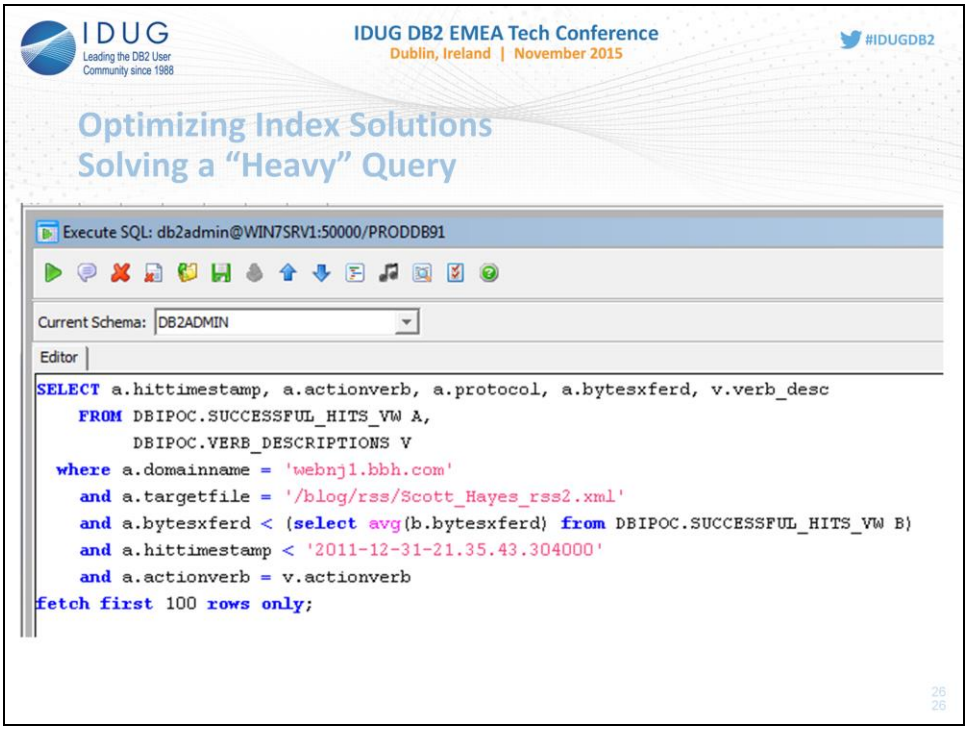
```
order by 7 ASC;
```



Sometimes you might think that db2advis uses drugs! It's awesome when db2advis recommends just one index that gives you a 99% cost reduction, but what about when db2advis wants you to create SEVERAL indexes? OMG!!!

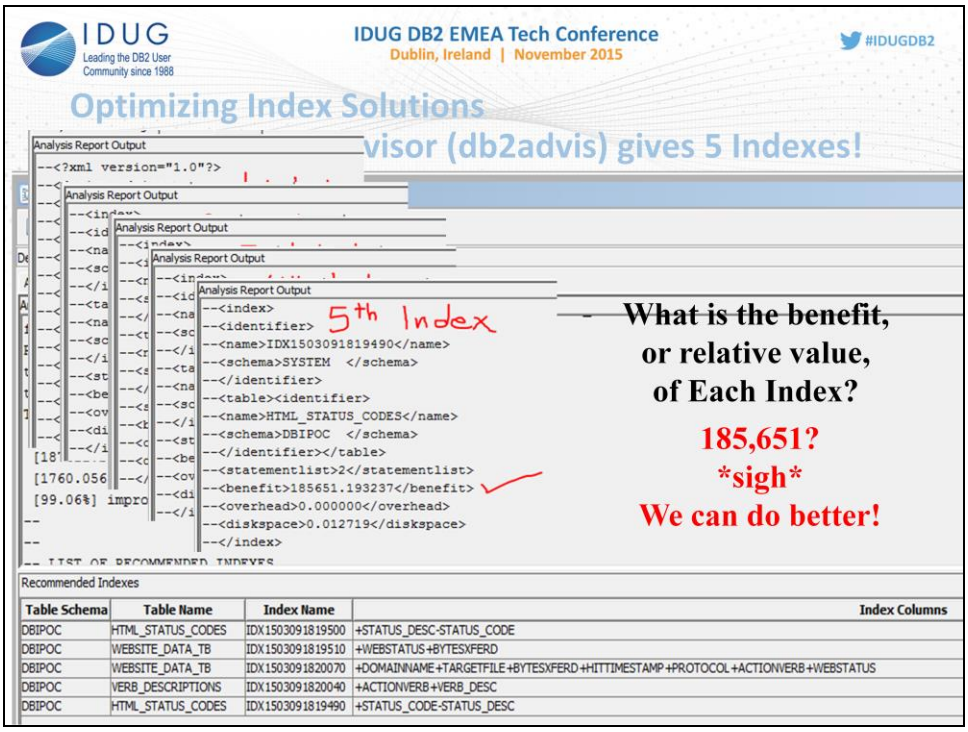
DB2Night Show #168 – Handout Update (This slide not in replay video files)

- The SQL examples in upcoming slides will help you do Index Benefit Analysis, with addition and subtraction, for a SQL workload of ONE SQL statement. If you want to analyze workloads of multiple SQL statements, the math is a bit different (more complicated and more time consuming) and beyond the scope of this presentation.
- **Instead of doing the calculations and analysis by hand, which, according to our polling survey, takes people several minutes to hours, why not just get DBI tools for DB2 LUW instead? You will have complete analysis for one, or multiple, SQL statement workloads within seconds!**



The "heavy" query:

```
SELECT a.hittimestamp, a.actionverb, a.protocol, a.bytesxferd, v.verb_desc
   FROM DBIPOC.SUCCESSFUL_HITS_VW A,
        DBIPOC.VERB_DESCRIPTIONS V
  where a.domainname = 'webnj1.bbh.com'
   and a.targetfile = '/blog/rss/Scott_Hayes_rss2.xml'
   and a.bytesxferd < (select avg(b.bytesxferd) from
DBIPOC.SUCCESSFUL_HITS_VW B)
   and a.hittimestamp < '2011-12-31-21.35.43.304000'
   and a.actionverb = v.actionverb
fetch first 100 rows only;
```



Results from DB2 9.7 FP3 on Windows…

The workload is in ADVISE_WORKLOAD with workload name: DBI_Analysis_Workload_1425925159390

found 1 statements in the ADVISE_WORKLOAD table

Recommending indexes...

total disk space needed for initial set [23.849] MB

total disk space constrained to [1212.059] MB

Trying variations of the solution set.

  5 indexes in current solution

 [187411.2500] timerons (without recommendations)

 [1760.0568] timerons (with current solution)

 [99.06%] improvement

--

--

-- LIST OF RECOMMENDED INDEXES

-- ===========================

-- index[1],   0.013MB

```
  CREATE INDEX "SYSTEM  "."IDX1503091819500" ON "DBIPOC
"."HTML_STATUS_CODES"
  ("STATUS_DESC" ASC, "STATUS_CODE" DESC) ALLOW REVERSE
  SCANS COLLECT SAMPLED DETAILED STATISTICS;
  COMMIT WORK ;
-- index[2],   19.657MB
  CREATE INDEX "SYSTEM  "."IDX1503091819510" ON "DBIPOC
"."WEBSITE_DATA_TB"
  ("WEBSTATUS" ASC, "BYTESXFERD" ASC) ALLOW REVERSE
  SCANS COLLECT SAMPLED DETAILED STATISTICS;
  COMMIT WORK ;
-- index[3],    4.153MB
  CREATE INDEX "SYSTEM  "."IDX1503091820070" ON "DBIPOC
"."WEBSITE_DATA_TB"
  ("DOMAINNAME" ASC, "TARGETFILE" ASC, "BYTESXFERD"
  ASC, "HITTIMESTAMP" ASC, "PROTOCOL" ASC, "ACTIONVERB"
  ASC, "WEBSTATUS" ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
  COMMIT WORK ;
-- index[4],    0.013MB
  CREATE INDEX "SYSTEM  "."IDX1503091820040" ON "DBIPOC
"."VERB_DESCRIPTIONS"
  ("ACTIONVERB" ASC, "VERB_DESC" ASC) ALLOW REVERSE
  SCANS COLLECT SAMPLED DETAILED STATISTICS;
  COMMIT WORK ;
-- index[5],    0.013MB
  CREATE INDEX "SYSTEM  "."IDX1503091819490" ON "DBIPOC
"."HTML_STATUS_CODES"
  ("STATUS_CODE" ASC, "STATUS_DESC" DESC) ALLOW REVERSE
  SCANS COLLECT SAMPLED DETAILED STATISTICS;
  COMMIT WORK ;
--
--
```

```
-- RECOMMENDED EXISTING INDEXES
-- ============================
--
--
-- UNUSED EXISTING INDEXES
-- ============================
-- ============================
--
-- ====ADVISOR DETAILED XML OUTPUT==========
-- ==(Benefits do not include clustering recommendations)==
--
--<?xml version="1.0"?>
--<design-advisor>
--<index>
--<identifier>
--<name>IDX1503091819500</name>
--<schema>SYSTEM  </schema>
--</identifier>
--<table><identifier>
--<name>HTML_STATUS_CODES</name>
--<schema>DBIPOC  </schema>
--</identifier></table>
--<statementlist>2</statementlist>
--<benefit>185651.193237</benefit>
--<overhead>0.000000</overhead>
--<diskspace>0.012719</diskspace>
--</index>
--<index>
--<identifier>
--<name>IDX1503091819510</name>
--<schema>SYSTEM  </schema>
--</identifier>
```

--<table><identifier>
--<name>WEBSITE_DATA_TB</name>
--<schema>DBIPOC </schema>
--</identifier></table>
--<statementlist>2</statementlist>
--<benefit>185651.193237</benefit>
--<overhead>0.000000</overhead>
--<diskspace>19.657250</diskspace>
--</index>
--<index>
--<identifier>
--<name>IDX1503091820070</name>
--<schema>SYSTEM </schema>
--</identifier>
--<table><identifier>
--<name>WEBSITE_DATA_TB</name>
--<schema>DBIPOC </schema>
--</identifier></table>
--<statementlist>2</statementlist>
--<benefit>185651.193237</benefit>
--<overhead>0.000000</overhead>
--<diskspace>4.153344</diskspace>
--</index>
--<index>
--<identifier>
--<name>IDX1503091820040</name>
--<schema>SYSTEM </schema>
--</identifier>
--<table><identifier>
--<name>VERB_DESCRIPTIONS</name>
--<schema>DBIPOC </schema>
--</identifier></table>

--<statementlist>2</statementlist>
--<benefit>185651.193237</benefit>
--<overhead>0.000000</overhead>
--<diskspace>0.012719</diskspace>
--</index>
--<index>
--<identifier>
--<name>IDX1503091819490</name>
--<schema>SYSTEM </schema>
--</identifier>
--<table><identifier>
--<name>HTML_STATUS_CODES</name>
--<schema>DBIPOC </schema>
--</identifier></table>
--<statementlist>2</statementlist>
--<benefit>185651.193237</benefit>
--<overhead>0.000000</overhead>
--<diskspace>0.012719</diskspace>
--</index>
--<statement>
--<statementnum>2</statementnum>
--<statementtext>
-- SELECT a.hittimestamp, a.actionverb, a.protocol, a.bytesxferd,
-- v.verb_desc   FROM DBIPOC.SUCCESSFUL_HITS_VW A,
--   DBIPOC.VERB_DESCRIPTIONS V   where a.domainname
-- = 'webnj1.bbh.com'   and a.targetfile = '/blog/rss/Scott_Hayes_rss2.xml'
--   and a.bytesxferd < (select avg(b.bytesxferd) from
-- DBIPOC.SUCCESSFUL_HITS_VW B)   and a.hittimestamp
-- < '2011-12-31-21.35.43.304000'   and a.actionverb
-- = v.actionverb fetch first 100 rows only
--</statementtext>
--<objects>

--<identifier>
--<name>HTML_STATUS_CODES</name>
--<schema>DBIPOC </schema>
--</identifier>
--<identifier>
--<name>VERB_DESCRIPTIONS</name>
--<schema>DBIPOC </schema>
--</identifier>
--<identifier>
--<name>WEBSITE_DATA_TB</name>
--<schema>DBIPOC </schema>
--</identifier>
--<identifier>
--<name>IDX1503091819490</name>
--<schema>SYSTEM </schema>
--</identifier>
--<identifier>
--<name>IDX1503091820040</name>
--<schema>SYSTEM </schema>
--</identifier>
--<identifier>
--<name>IDX1503091820070</name>
--<schema>SYSTEM </schema>
--</identifier>
--<identifier>
--<name>IDX1503091819510</name>
--<schema>SYSTEM </schema>
--</identifier>
--<identifier>
--<name>IDX1503091819500</name>
--<schema>SYSTEM </schema>
--</identifier>

--</objects>

--<benefit>185651.193237</benefit>

--<frequency>1</frequency>

--</statement>

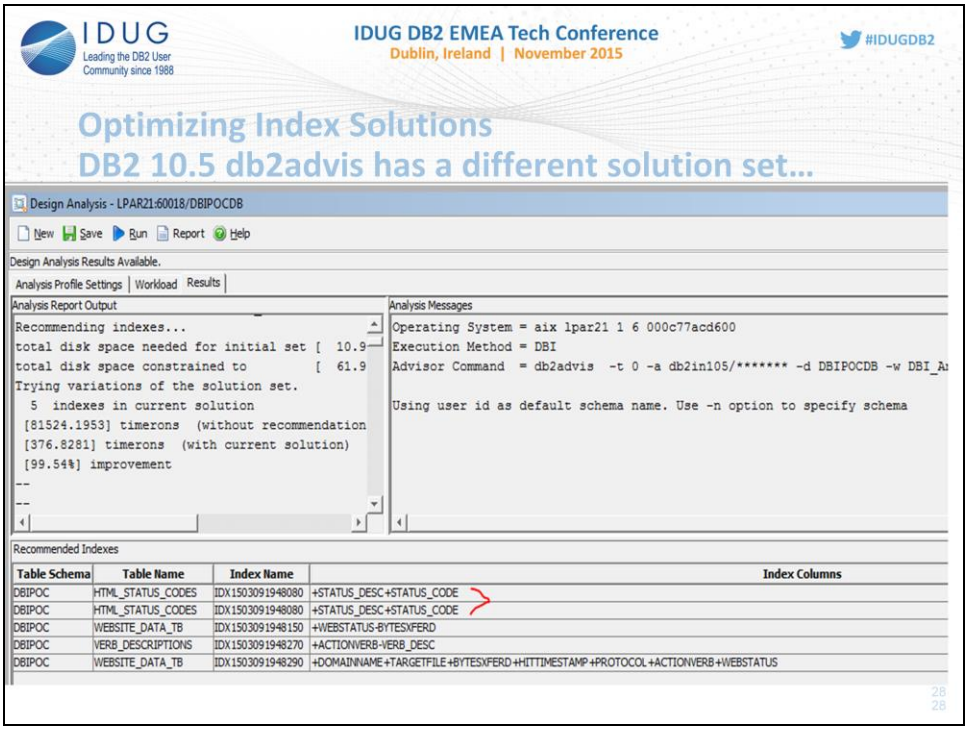
--</design-advisor>

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

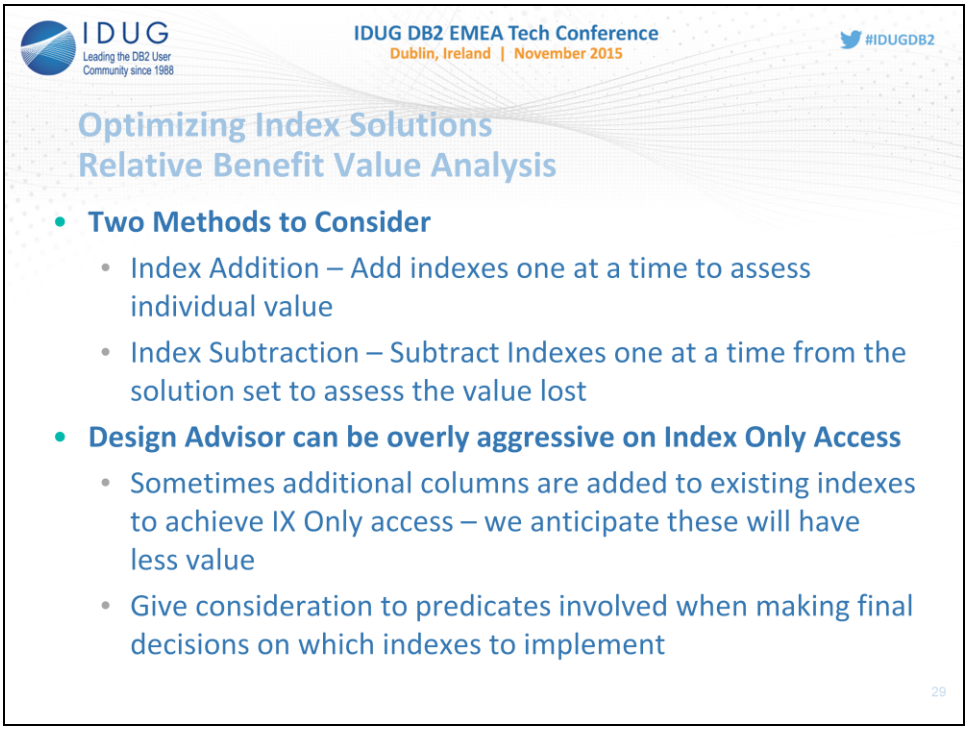
--

326 solutions were evaluated by the advisor

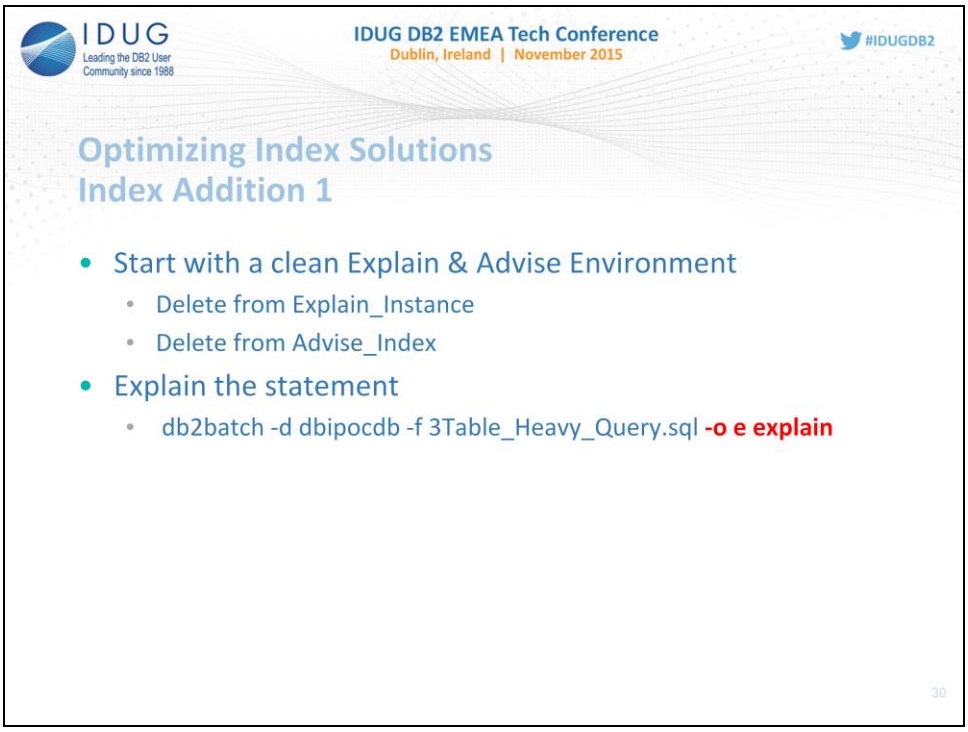
DB2 Workload Performance Advisor tool is finished.



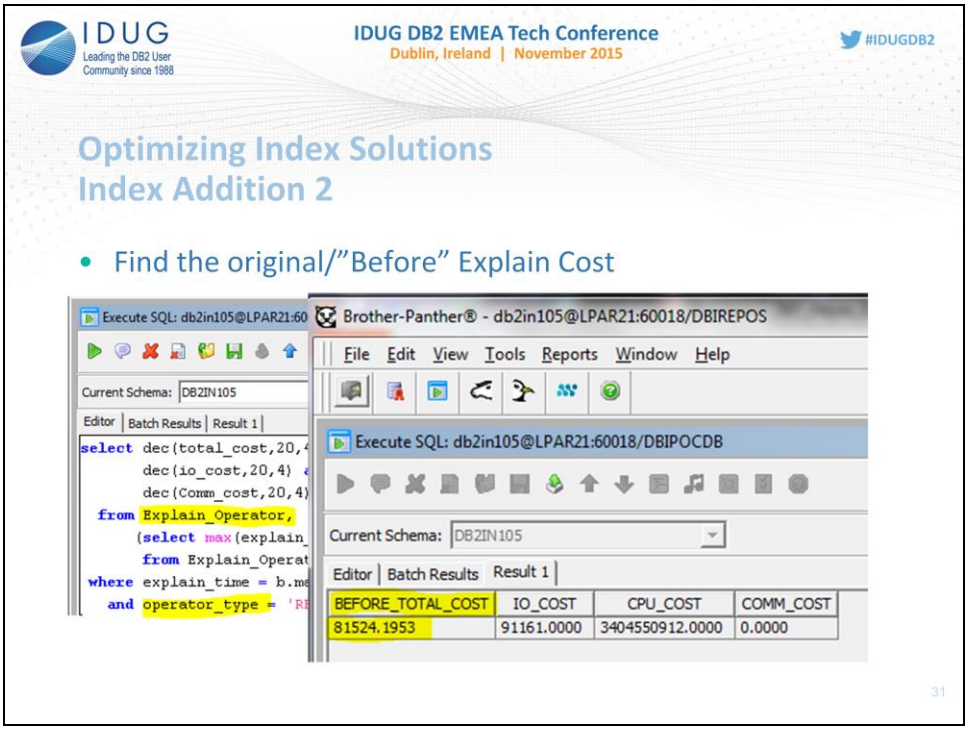
Results from DB2 10.5.2 on AIX 6.1. We'll proceed with working in this environment because I like AIX better!



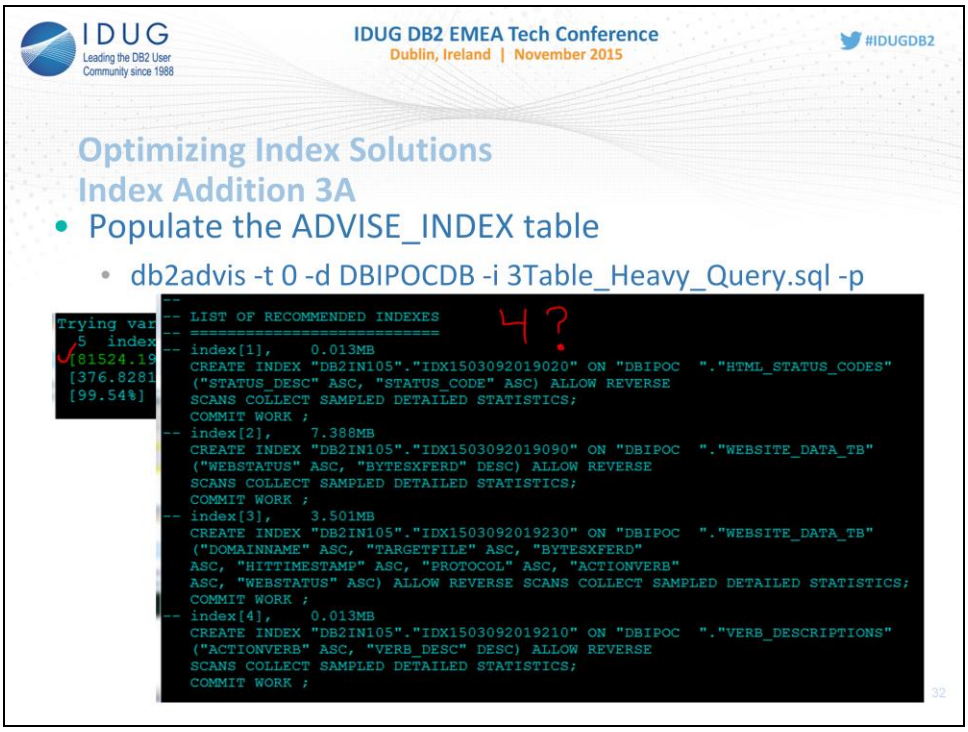
The IBM Design Advisor claims to have evaluated 326 solutions to arrive at the proposed solution set. Presuming disk storage is tight (when is it not? (rhetorical)) and time is limited, we will next look at the steps required to determine the relative weighted value of each proposed index.



The "-o e explain" option causes db2batch to Explain the statements but NOT run them!

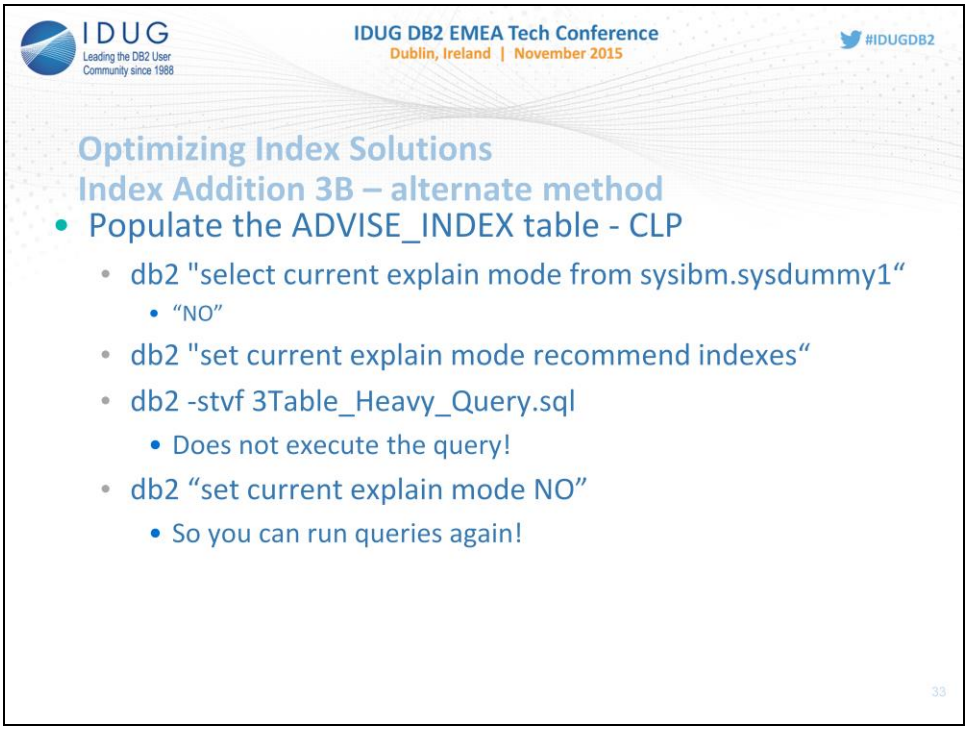



```
select dec(total_cost,20,4) as before_total_cost,
       dec(io_cost,20,4) as io_cost, dec(CPU_cost,20,4) as cpu_cost,
       dec(Comm_cost,20,4) as comm_cost
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR ;
```

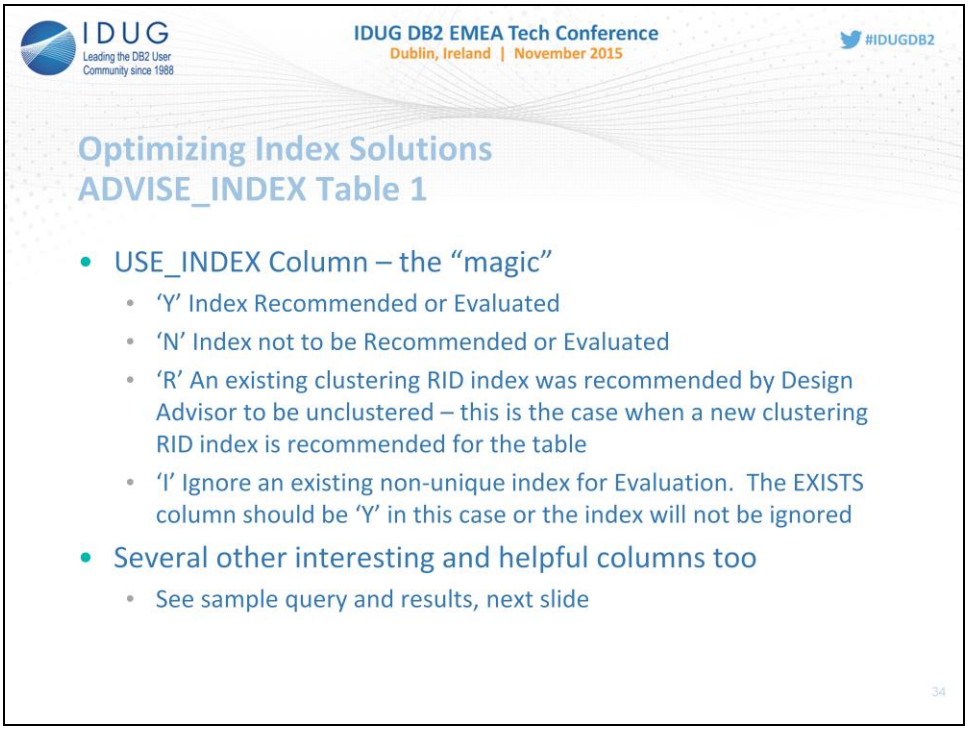


163 Solutions were evaluated by db2advis.

db2advis -t 0 -d DBIPOCDB -i 3Table_Heavy_Query.sql



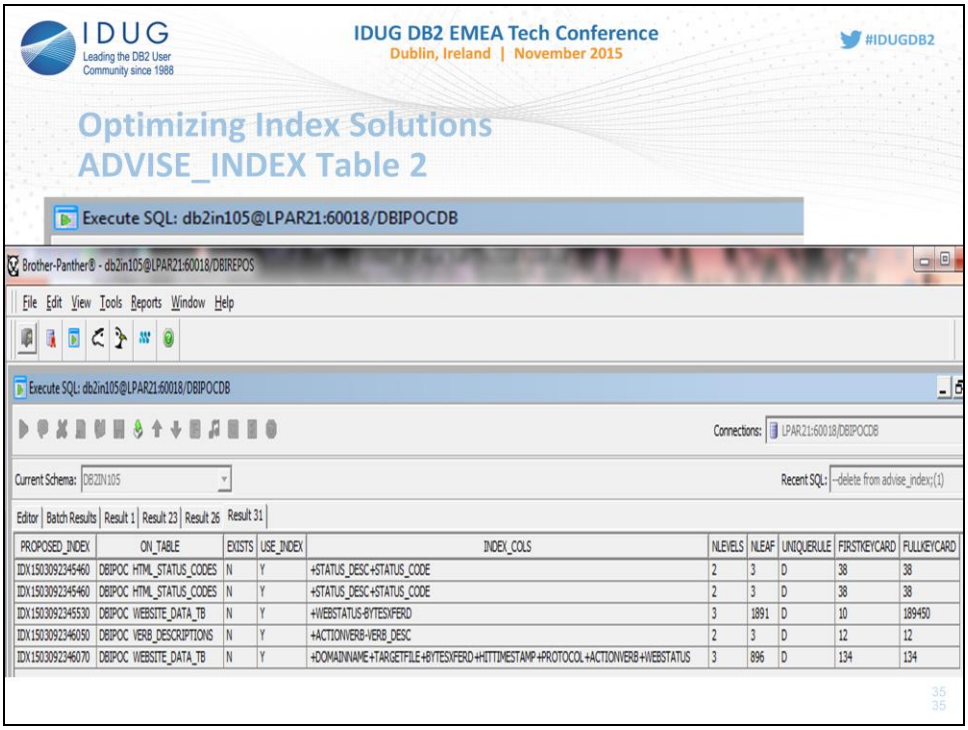
For this type of index benefit analysis, I actually prefer this CLP method over db2advis, though I do appreciate how db2advis provides nicely summarized outputs.



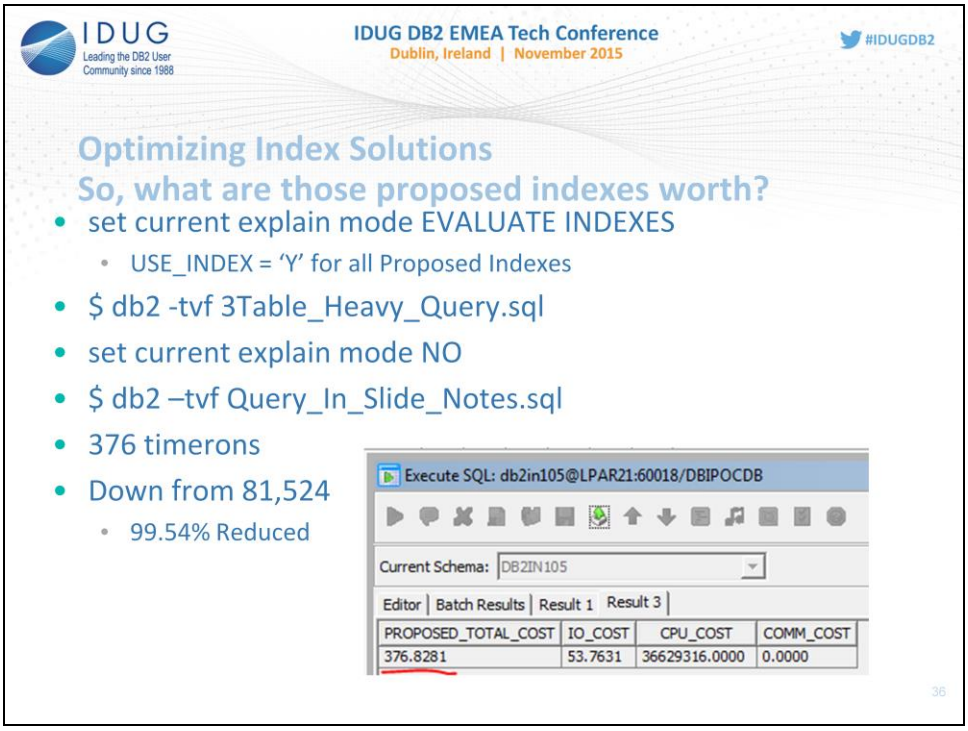
Unclustered is a valid word, but spell check wanted it changed to uncluttered. Funny.

Learn about the ADVISE_INDEX table:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0002417.html?cp=SSEPGG_10.5.0%2F2-12-13-0&lang=en

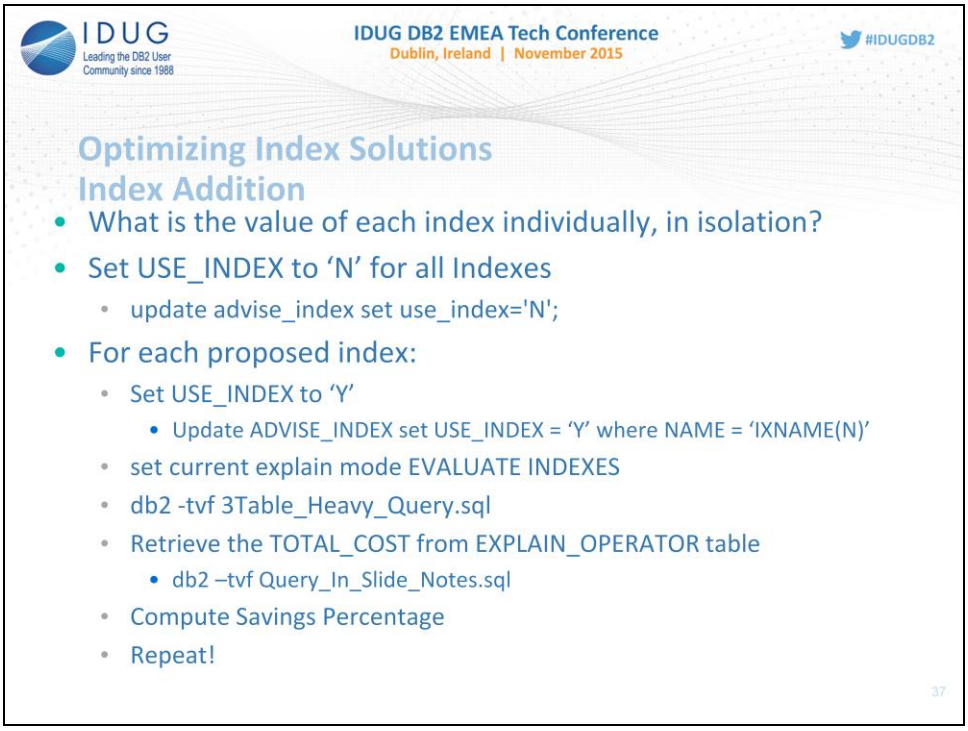



```
select varchar(name,20) as PROPOSED_INDEX,
       concat(tbcreator, tbname) as ON_TABLE,
       EXISTS, USE_INDEX, varchar(colnames,80) as INDEX_COLS,
       NLEVELS, NLEAF, UNIQUERULE, FIRSTKEYCARD,
FULLKEYCARD
   from advise_index;
```

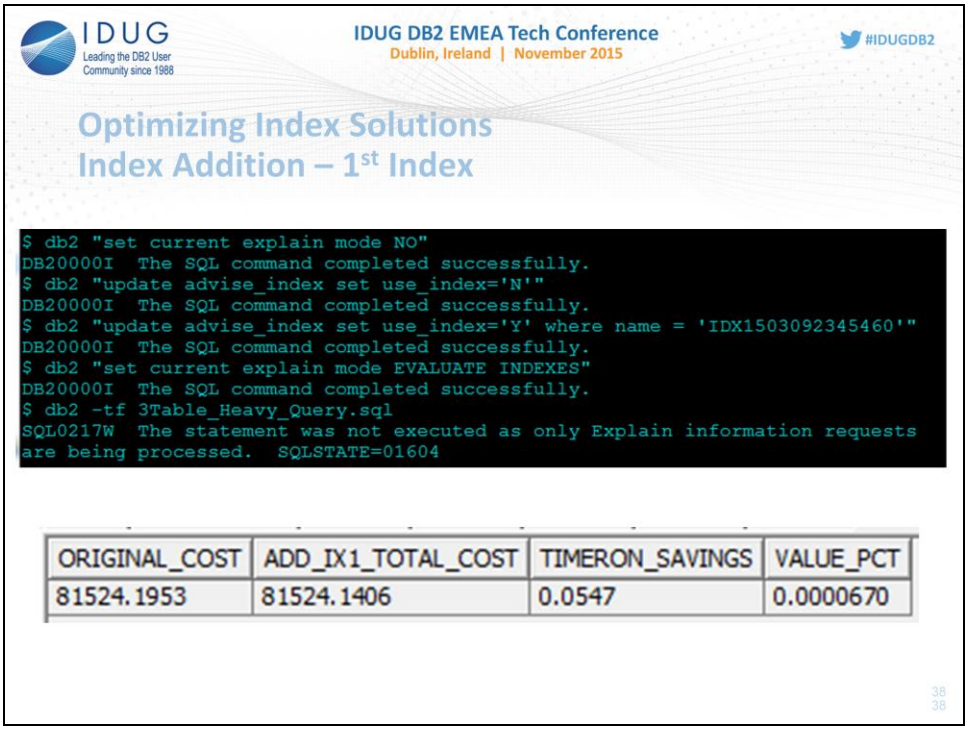


Find the new query cost with the proposed indexes:

```
select dec(total_cost,20,4) as proposed_total_cost,
       dec(io_cost,20,4) as io_cost, dec(CPU_cost,20,4) as cpu_cost,
       dec(Comm_cost,20,4) as comm_cost
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```

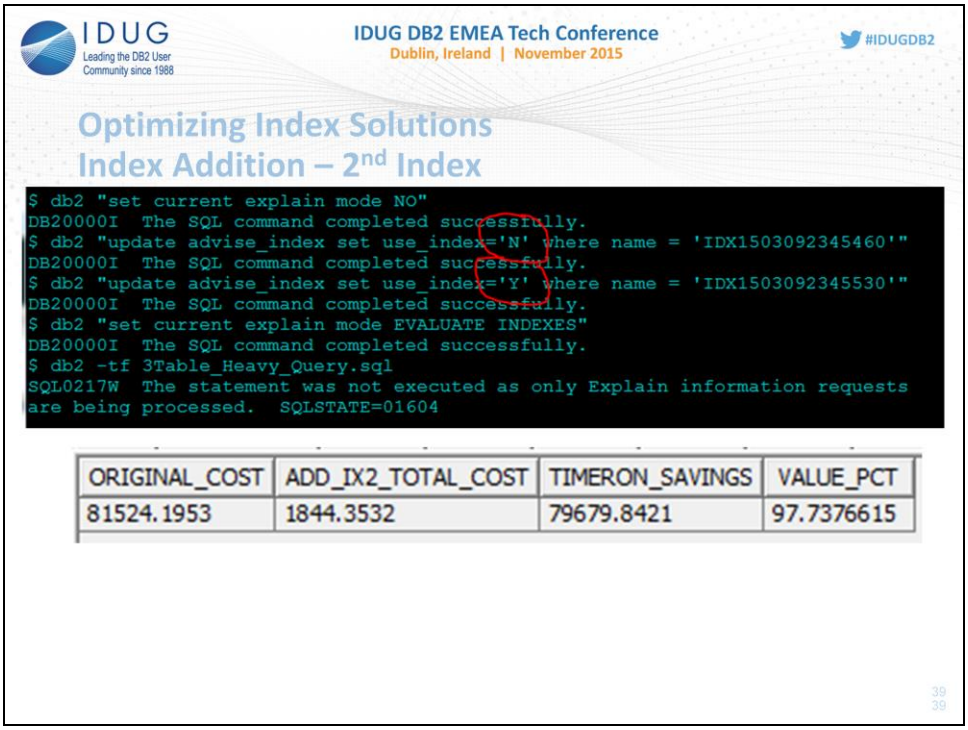



```
select dec(total_cost,20,4) as add_IXNAME_total_cost
from Explain_Operator,
     (select max(explain_time) as maxtime
      from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



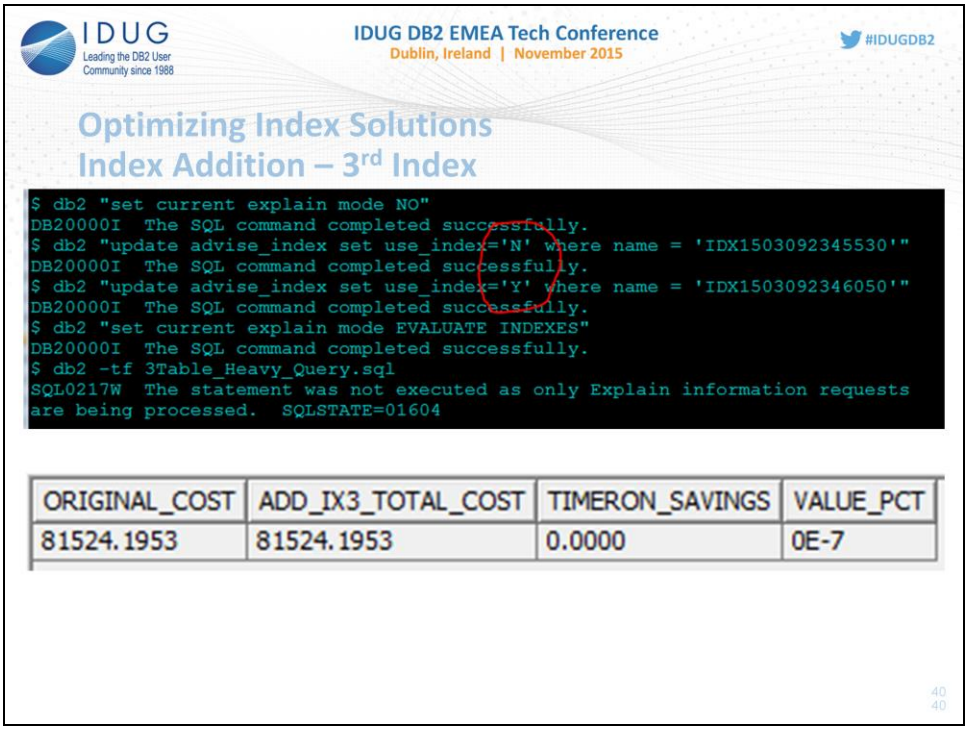
Query to find cost savings:

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as add_IX1_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as value_pct
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



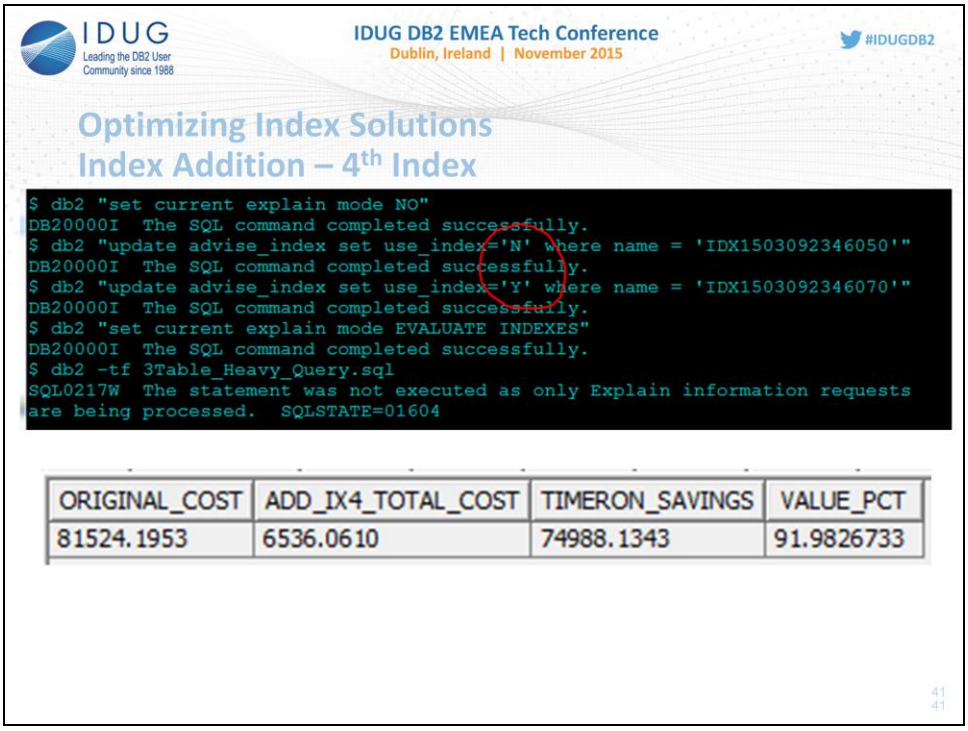
Query to find cost savings:

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as add_IX2_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as value_pct
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



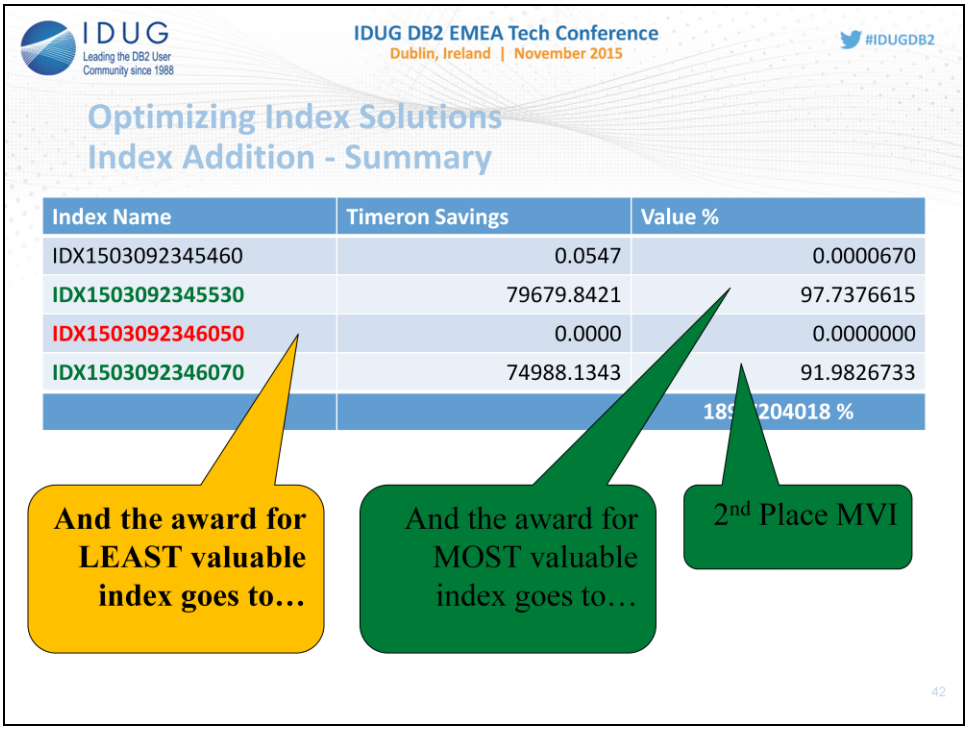
Query to find cost savings:

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as add_IX3_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as value_pct
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```

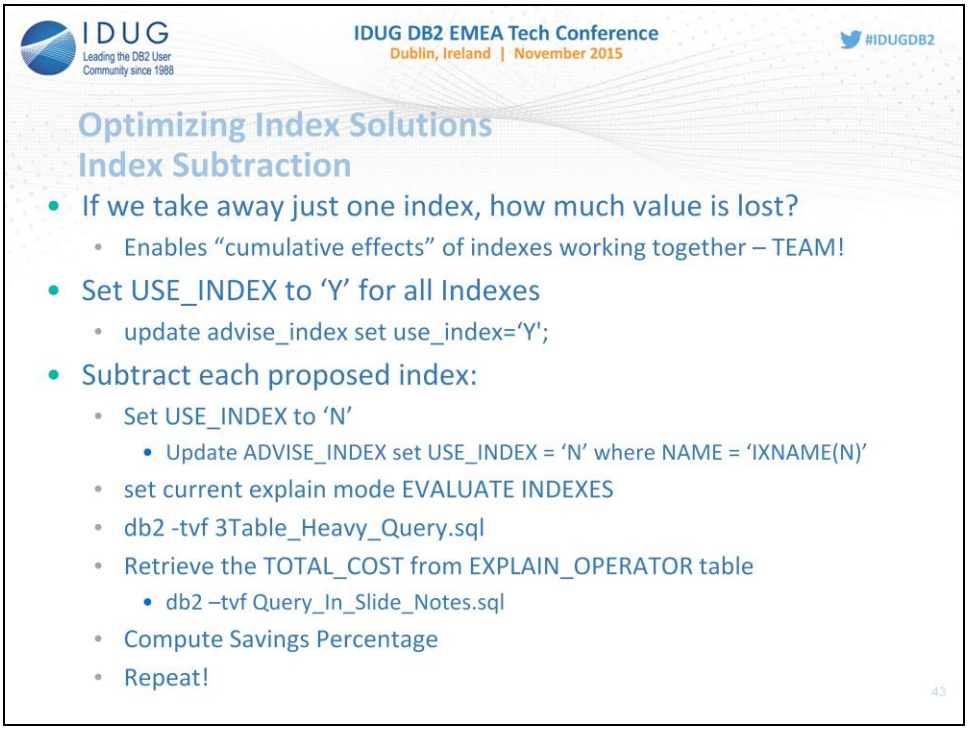


Query to find cost savings:

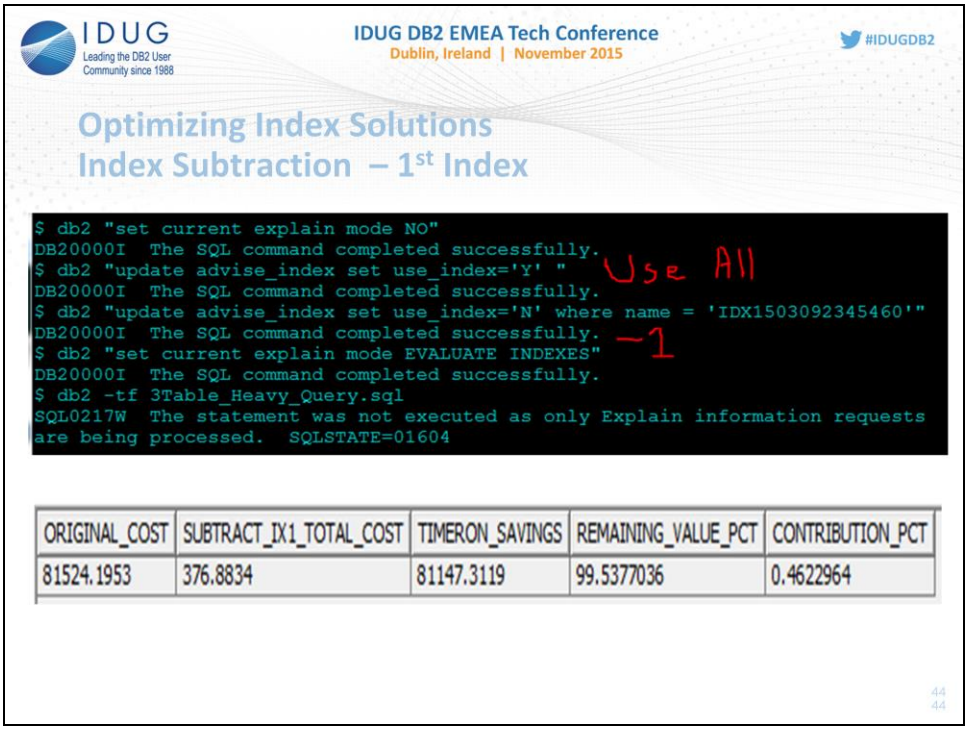
```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as add_IX4_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as value_pct
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



By looking at proposed indexes in isolation, optimizer plans could change significantly, and some indexes may stand out as having very significant value. Wouldn't it be nice if the cost could be reduced by 189% ?!?!? We'd be getting back FREE resources from DB2 just by running the query! HA!

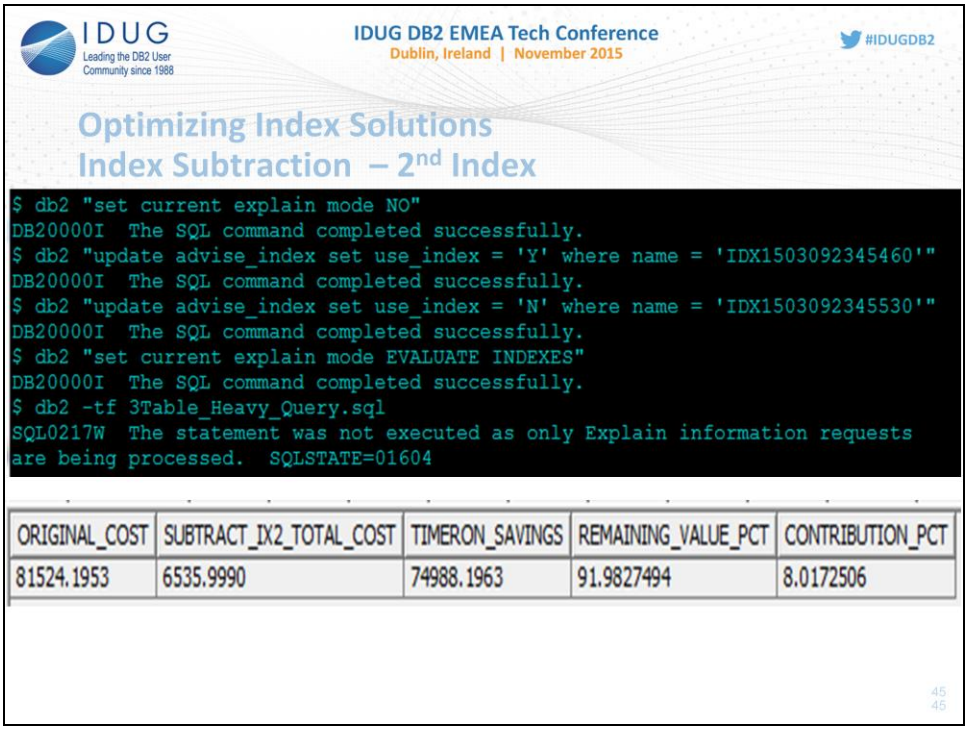



```
select dec(total_cost,20,4) as subtract_IXNAME_total_cost
from Explain_Operator,
     (select max(explain_time) as maxtime
      from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```

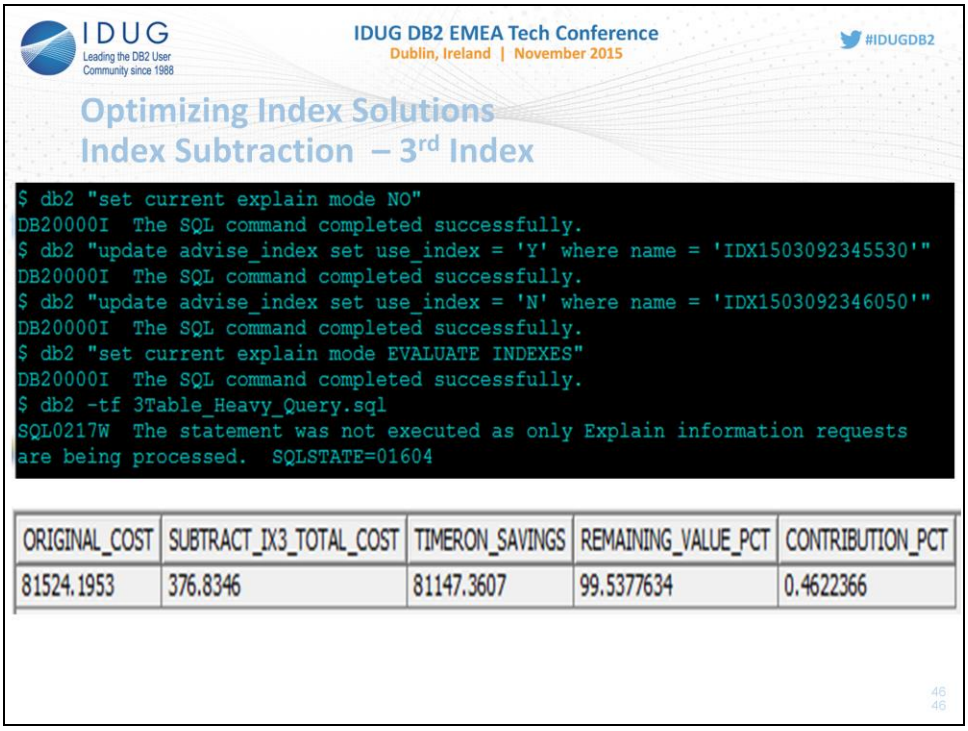


Query to find cost savings:

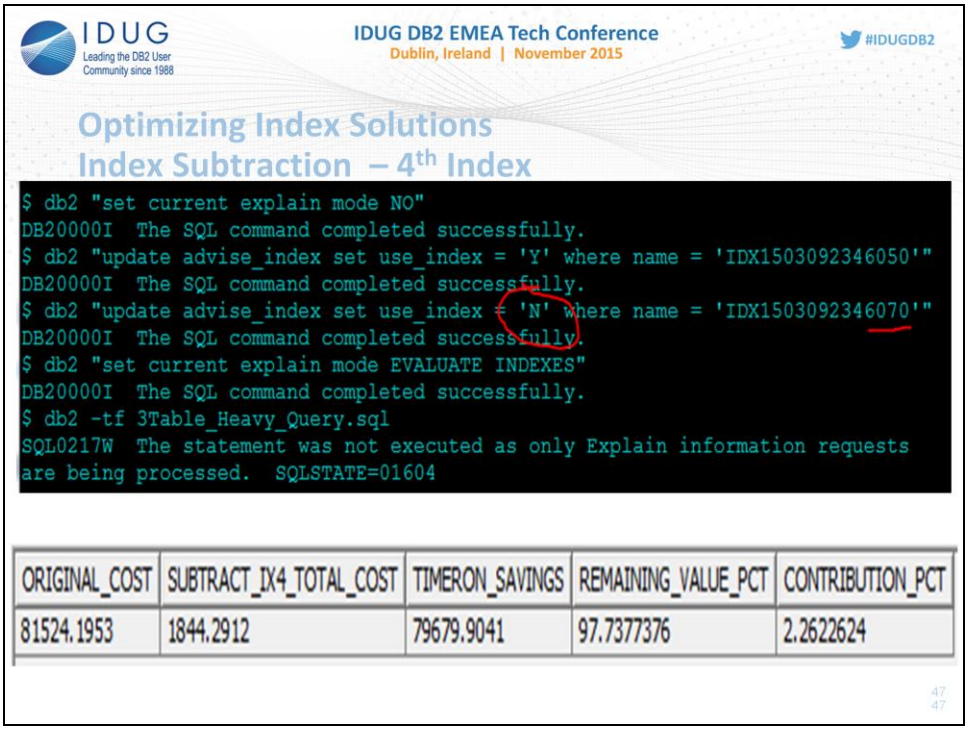
```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as subtract_IX1_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as
Remaining_value_pct,
       100.0 - (((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 ) as
Contribution_PCT
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



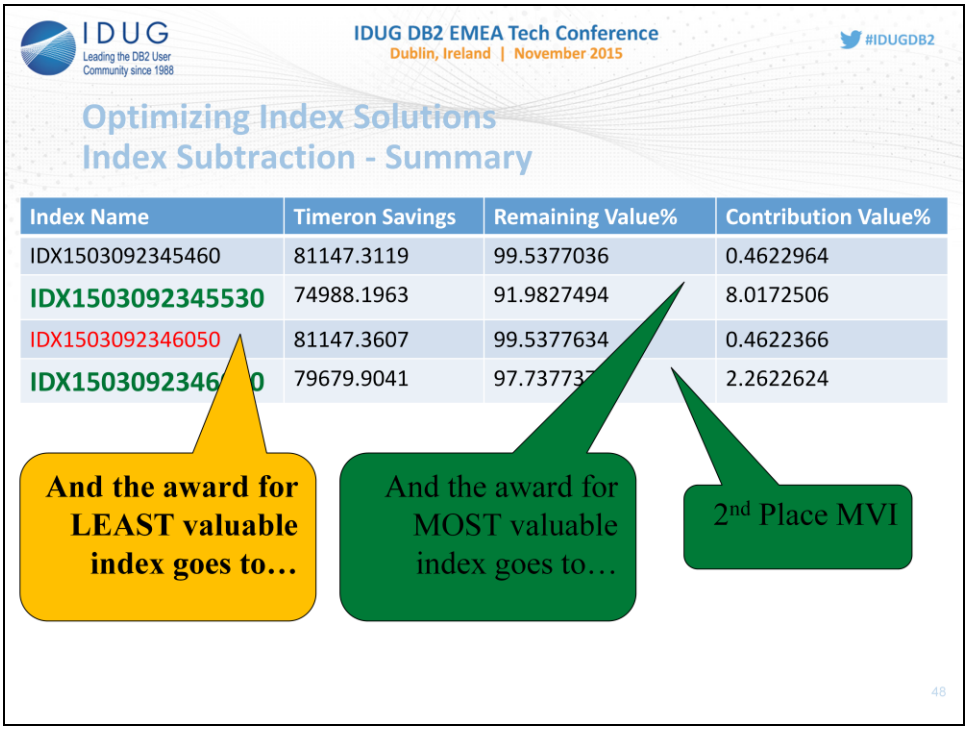

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as subtract_IX2_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as
Remaining_value_pct,
       100.0 - (((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 ) as
Contribution_PCT
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



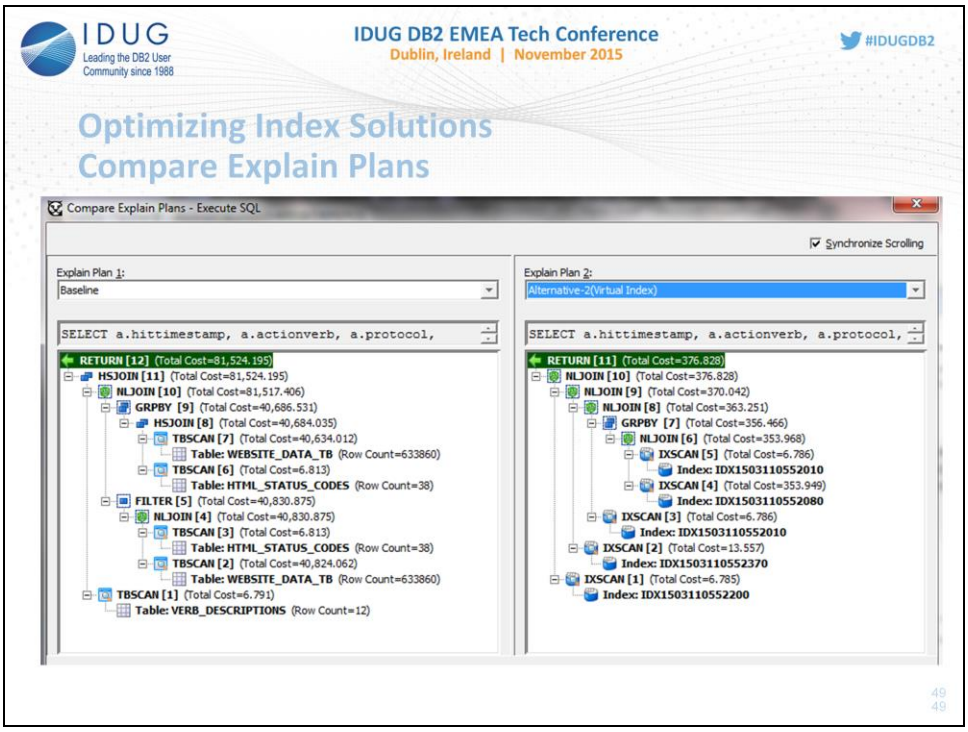

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as subtract_IX3_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as
Remaining_value_pct,
       100.0 - (((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 ) as
Contribution_PCT
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



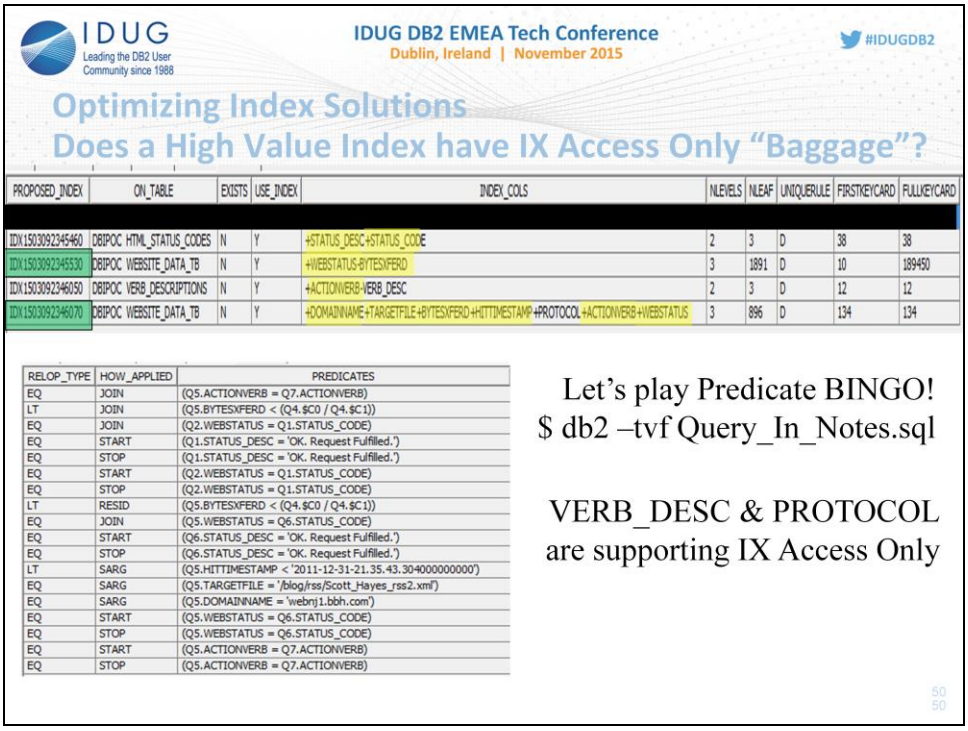

```
select 81524.1953 as Original_Cost,
       dec(total_cost,20,4) as subtract_IX3_total_cost,
       81524.1953 - dec(total_cost,20,4) as timeron_savings,
       ((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 as
Remaining_value_pct,
       100.0 - (((81524.1953 - dec(total_cost,20,4)) * 100.0) / 81524.1953 ) as
Contribution_PCT
  from Explain_Operator,
      (select max(explain_time) as maxtime
       from Explain_Operator) as b
 where explain_time = b.maxtime
   and operator_type = 'RETURN'  with UR;
```



We've now separated the "mice" from the "men". By addition and by subtraction, we know which indexes are the most valuable, providing the most benefit, and which are the least valuable. But wait, there's more…



With all of the recommended indexes created, you will note that there are NO TBSCAN operations. Recommend Indexes and/or db2advis has generated Index Only solutions.



Query to look at EXPLAIN_PREDICATES table:

```
select a.relop_type, a.how_applied,
  varchar(a.predicate_text,100) as predicates
  from explain_predicate a
  where a.explain_time = (select max(b.explain_time) from explain_predicate
b);
```

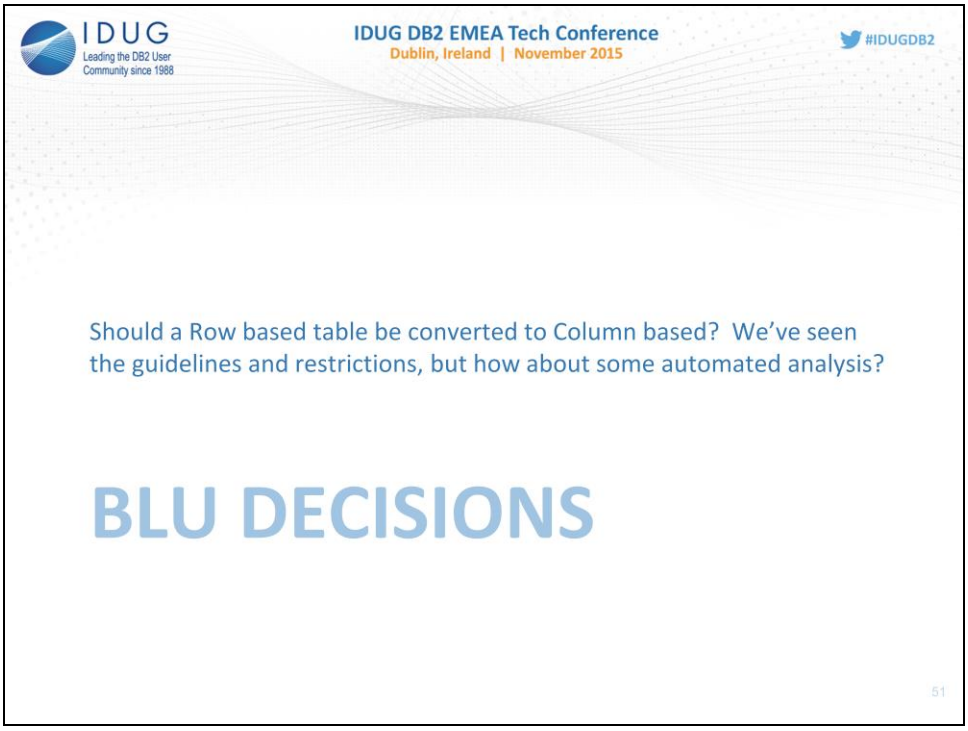
VERB_DESC in Index IDX1503092346050 is giving IX Access Only but no predicate value

PROTOCOL in Index IDX1503092346070 is giving IX Access Only but no predicate value

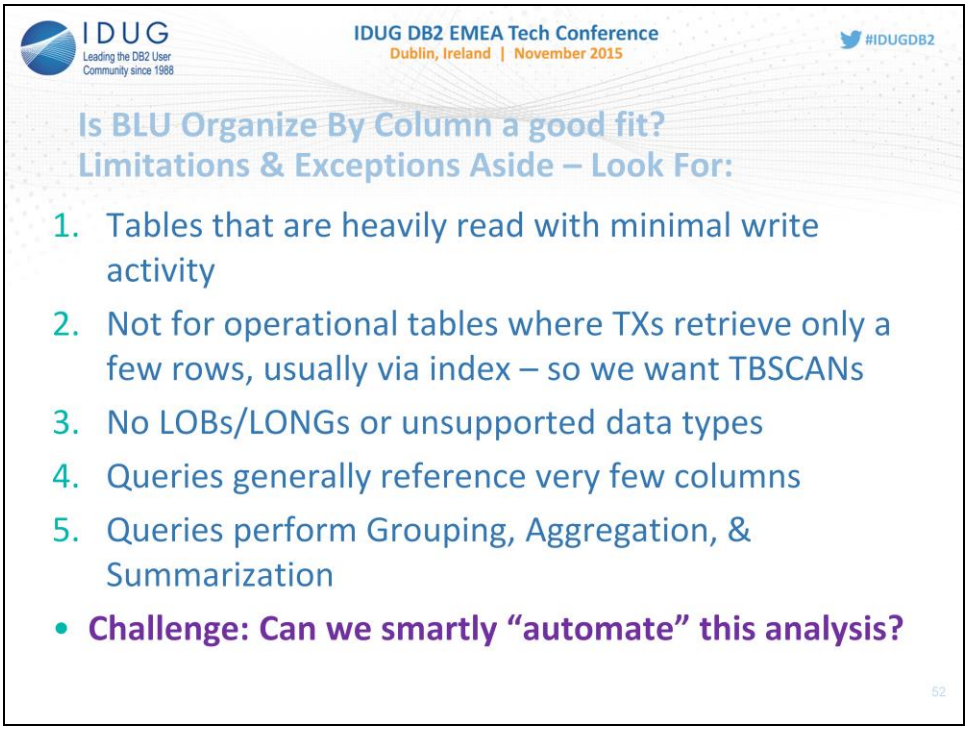
Reserve IX Access Only for HEAVY SQL that is FREQUENTLY executed within important transactions.

Sometimes several extra columns might be added to an index to achieve IX Access Only – they could be, and probably should be, omitted, unless their inclusion also helps to increase the IX FULLKEYCARD (FULLKEYCARD

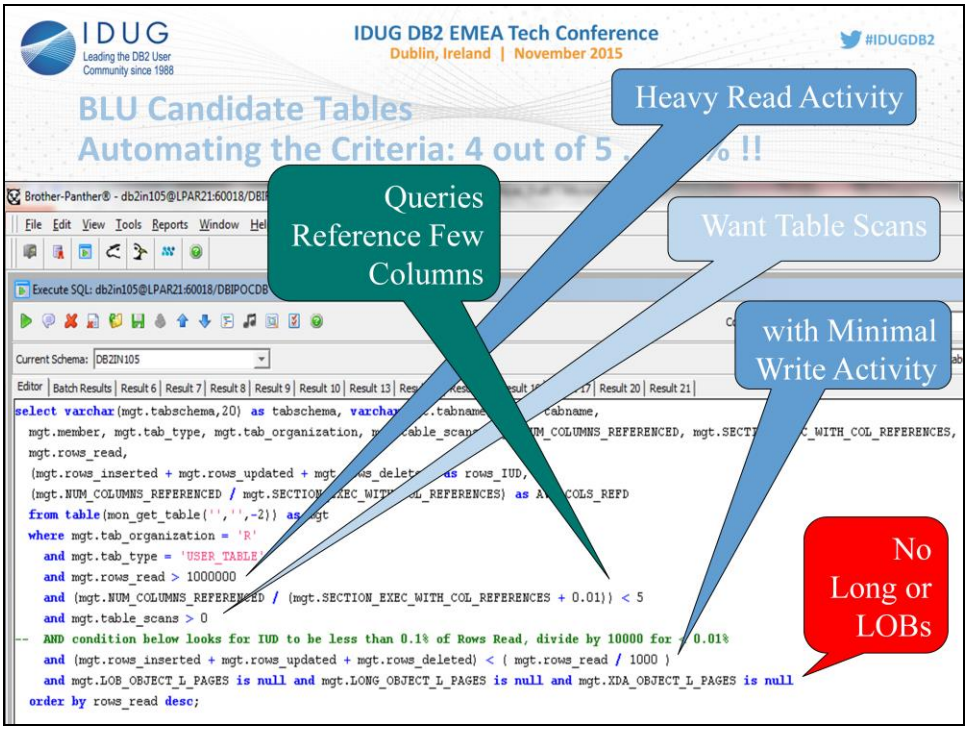
ideally > 5% of TBCARD)



Feeling BLU? Have success stories or marketing propaganda peaked your interest? Do you have a Data Warehouse? Let's see if we can find some candidate tables for Column Organization…

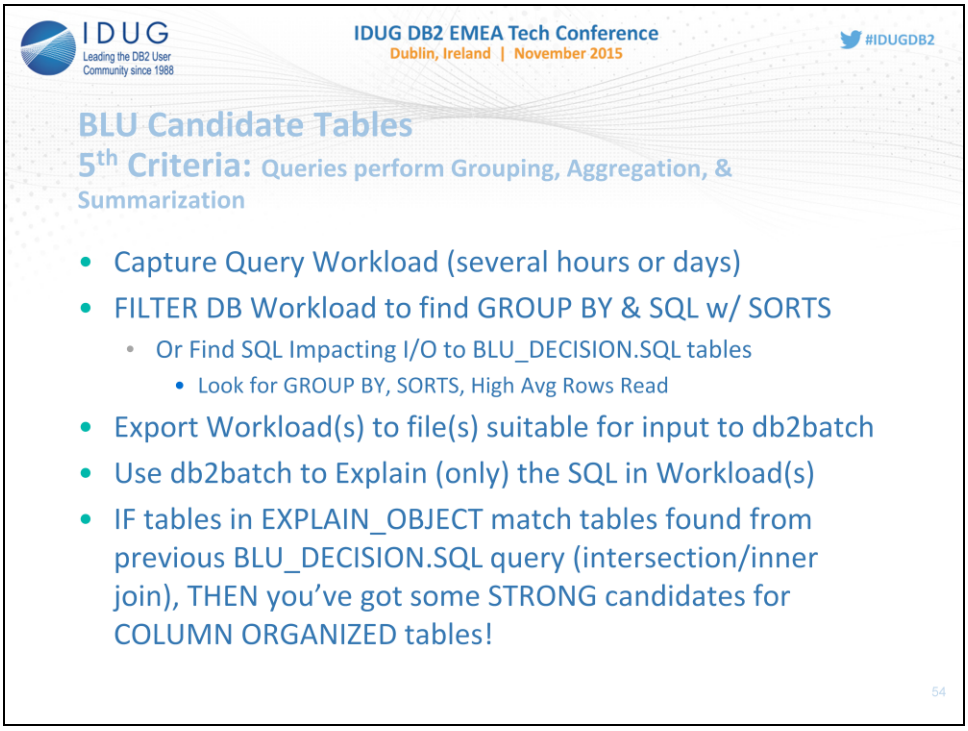


I love a good challenge!

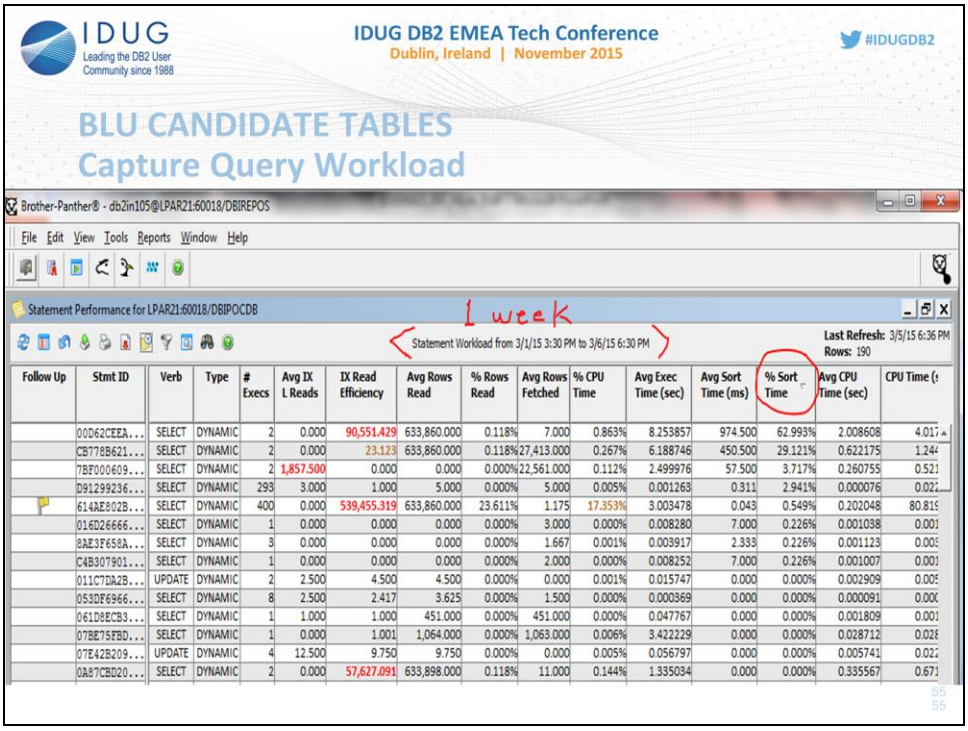



```
select varchar(mgt.tabschema,20) as tabschema, varchar(mgt.tabname,20) as
tabname,
  mgt.member, mgt.tab_type, mgt.tab_organization, mgt.table_scans,
mgt.NUM_COLUMNS_REFERENCED,
mgt.SECTION_EXEC_WITH_COL_REFERENCES,
  mgt.rows_read,
  (mgt.rows_inserted + mgt.rows_updated + mgt.rows_deleted) as rows_IUD,
  (mgt.NUM_COLUMNS_REFERENCED /
mgt.SECTION_EXEC_WITH_COL_REFERENCES) as AVG_COLS_REFD
  from table(mon_get_table('','',-2)) as mgt
  where mgt.tab_organization = 'R'
    and mgt.tab_type = 'USER_TABLE'
    and mgt.rows_read > 1000000
    and (mgt.NUM_COLUMNS_REFERENCED /
(mgt.SECTION_EXEC_WITH_COL_REFERENCES + 0.01)) < 5
    and mgt.table_scans > 0
--  AND condition below looks for IUD to be less than 0.1% of Rows Read,
divide by 10000 for < 0.01%
```

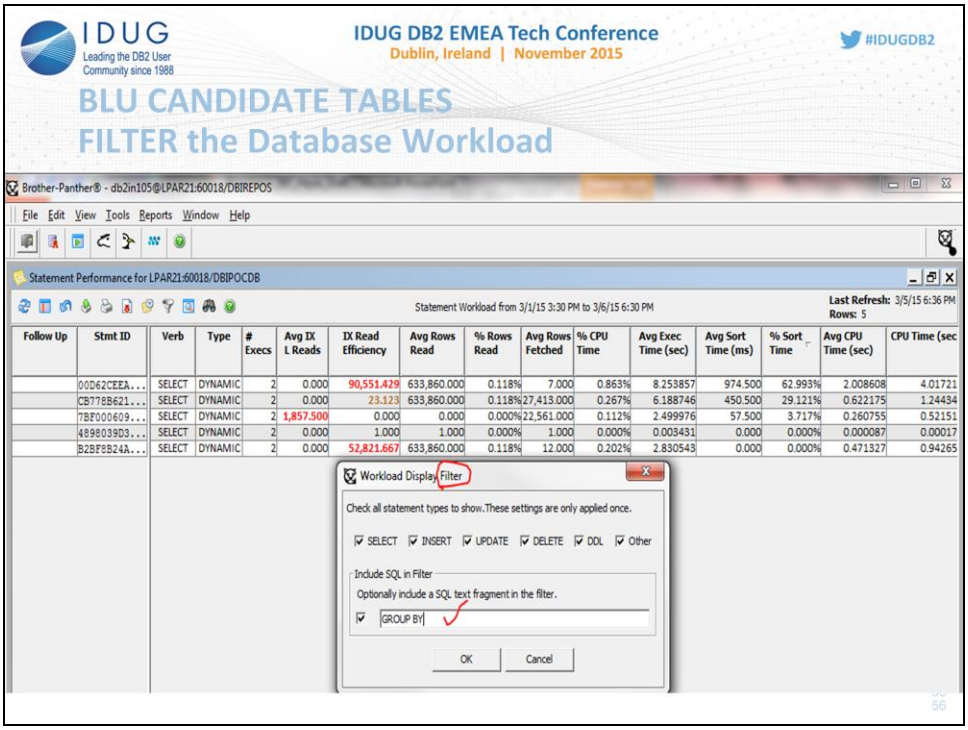
```
    and (mgt.rows_inserted + mgt.rows_updated + mgt.rows_deleted) < (
mgt.rows_read / 1000 )
    and mgt.LOB_OBJECT_L_PAGES is null and mgt.LONG_OBJECT_L_PAGES is
null and mgt.XDA_OBJECT_L_PAGES is null
  order by rows_read desc;
```



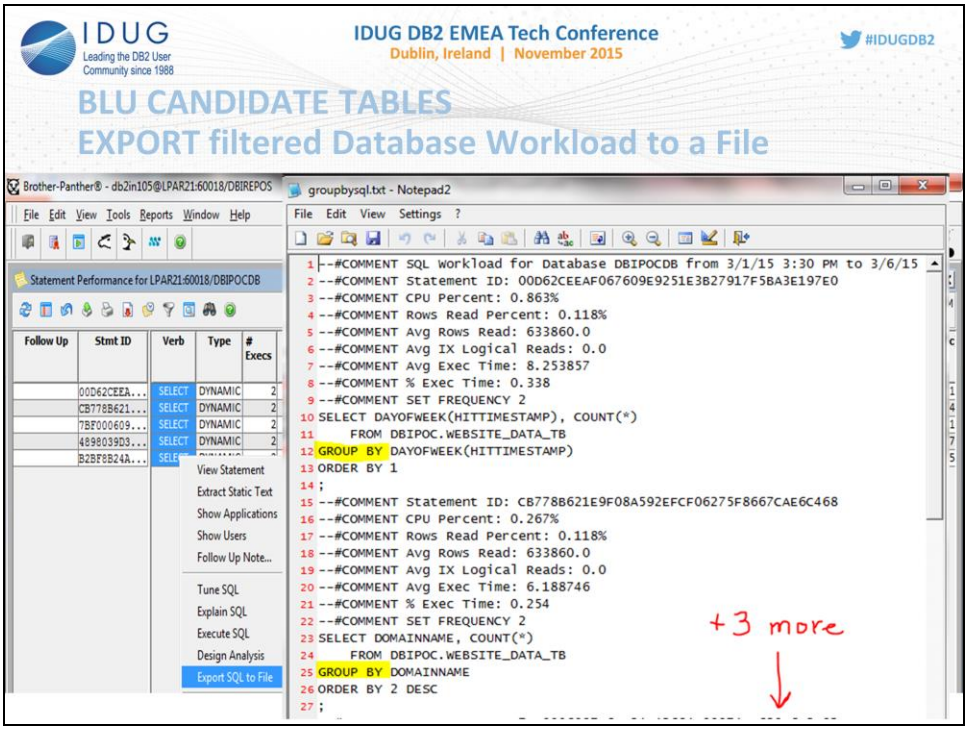
Where can we find a relationship between tables and SQL workloads? EXPLAIN! In particular, EXPLAIN_OBJECT table will tell you about tables, indexes, and other objects used to run a SQL query. This slide describes a methodology for discovering tables that are "victims" of queries that perform grouping, aggregation, and summarization. As always, individual results may vary.



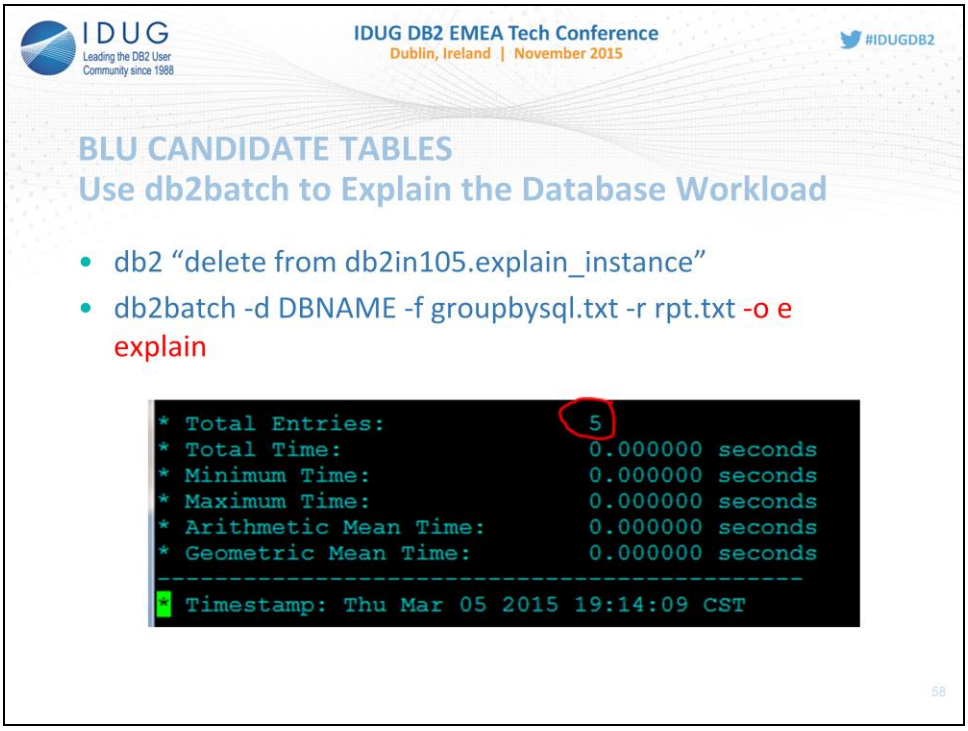
There are commercially available tools that can help you capture and manage SQL workloads, or with greater difficulty, you could capture SQL from db2pd, db2top, or queries to MON_GET_PACKAGE_CACHE. For purposes of demonstrating the METHOD of this process, DBI's Brother-Panther is illustrated.



DBI's Brother-Panther allows you to filter workloads by types of SQL or SQL containing certain strings. Alternatively, in a command line interface, you might pass the SQL workload through grep or equivalent OS command.



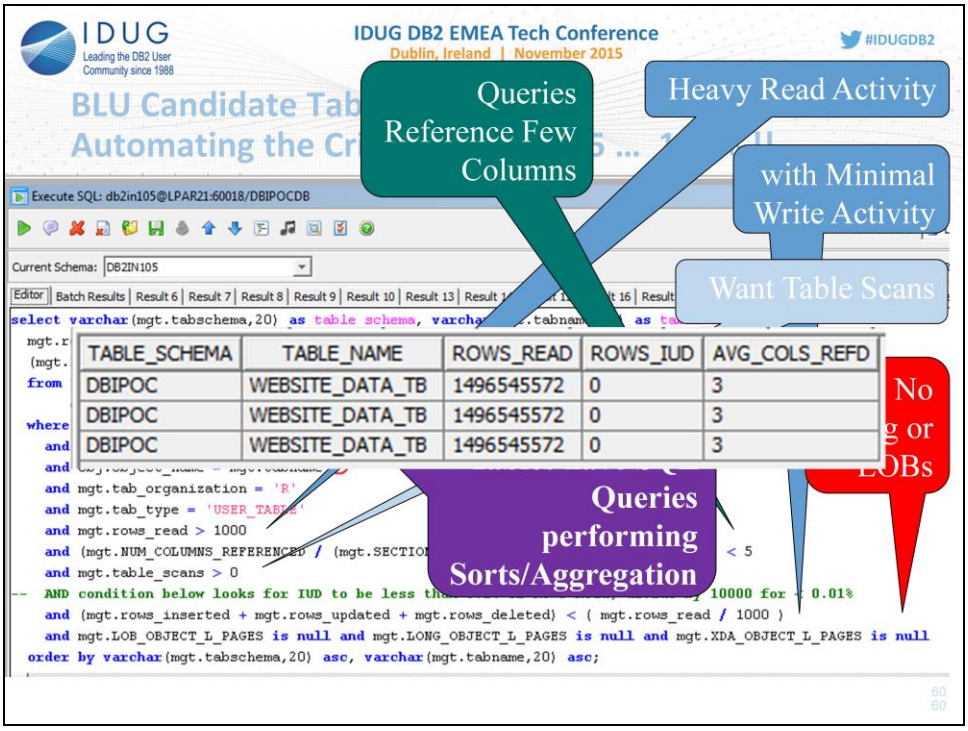
The filtered workload (all statements containing GROUP BY) is exported to a flat text file that is suitable for input to db2batch or db2advis. You should notice that there are also liberal comments that document the performance attributes of the exported statements.



Begin by clearing out the Explain tables. Delete to Explain_Instance does a cascading delete to other Explain tables.

The "-o e explain" option instructs db2batch to Explain the statements but not run them!

This is exciting!



select varchar(mgt.tabschema,20) as table_schema, varchar(mgt.tabname,20) as table_name,

mgt.rows_read, (mgt.rows_inserted + mgt.rows_updated + mgt.rows_deleted) as rows_IUD,

(mgt.NUM_COLUMNS_REFERENCED / mgt.SECTION_EXEC_WITH_COL_REFERENCES) as AVG_COLS_REFD

from table(mon_get_table('','',-2)) as mgt,

explain_object as obj

where obj.object_type = 'TA'

and obj.object_schema = mgt.tabschema

and obj.object_name = mgt.tabname

and mgt.tab_organization = 'R'

and mgt.tab_type = 'USER_TABLE'

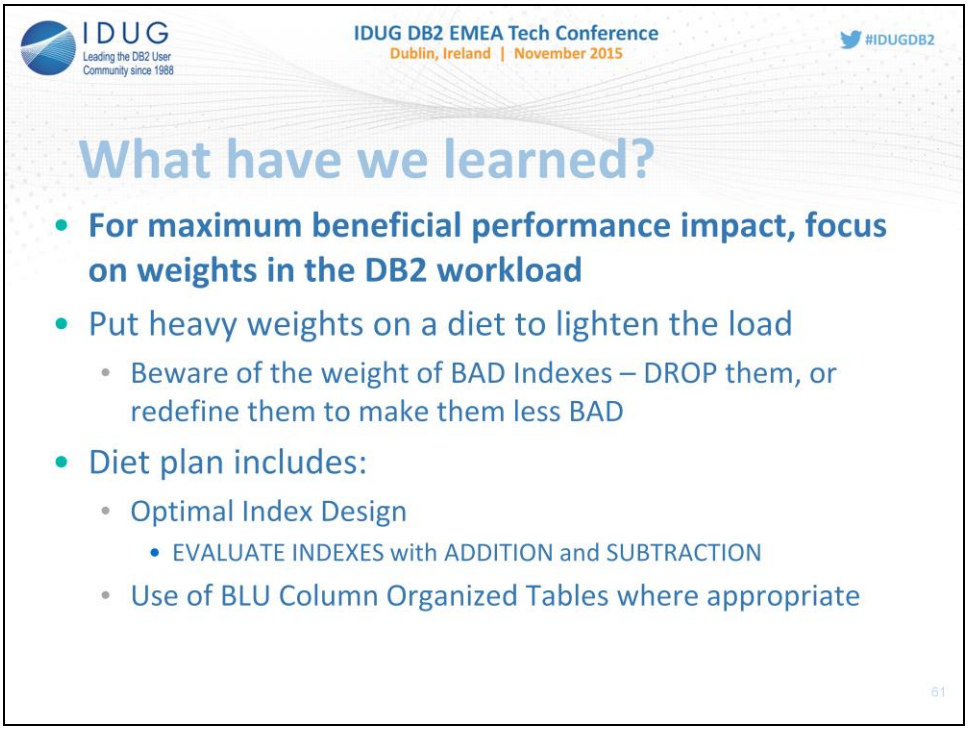
and mgt.rows_read > 1000

and (mgt.NUM_COLUMNS_REFERENCED / (mgt.SECTION_EXEC_WITH_COL_REFERENCES + 0.01)) < 5

and mgt.table_scans > 0

-- AND condition below looks for IUD to be less than 0.1% of Rows Read,

```
divide by 10000 for < 0.01%
   and (mgt.rows_inserted + mgt.rows_updated + mgt.rows_deleted) < (
mgt.rows_read / 1000 )
   and mgt.LOB_OBJECT_L_PAGES is null and mgt.LONG_OBJECT_L_PAGES is
null and mgt.XDA_OBJECT_L_PAGES is null
  order by varchar(mgt.tabschema,20) asc, varchar(mgt.tabname,20) asc;
```



Learn more from this IDUG blog: http://www.idug.org/p/bl/ar/blogaid=351

Learn more from the DB2 LUW Performance blogs: http://www.dbisoftware.com/blog/

The end! Follow me on twitter at @srhayes! Follow DBI at @dbisoftware