

Platform: z/OS

## Tune SQL like an expert in DB2 UDB for z/OS V8

*Terry Purcell*

*IBM Silicon Valley Lab*

**Session:** G7

**Date/Time:** Tuesday May 24<sup>th</sup>, 3:30 – 4:40pm



## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process

Note: References to Visual Explain are Visual Explain for DB2 UDB for z/OS V8 only.  
This is a free tool downloadable from IBM at:

<http://www-306.ibm.com/software/data/db2/zos/osc/ve/>

## Finding the problem query

- Reactively
  - User complaint
  - Trace output
  - Etc...or any other traditional method
- Proactively
  - Using Visual Explain to
    - Search for previously explained static SQL
      - Apply cost (estimate) or access path filters
    - View dynamic statement cache
      - Reactive, but before user complaint!!
      - Apply filters based upon query execution

## VE Static SQL Input – Cost Filters

Filter name

x

Only list explainable statements  List all static statements

New Save Save As Delete

**Packages** Plans Cost/object Filters Access Path Filters Summary

Use cost filter (Which requires that the DSN\_STATEMENT\_TABLE have the corresponding records)

Use object filter (Which requires that the plan\_table have the corresponding records)

Select the cost and object criteria for the static SQL statements that you want to list

Name	Operator	Value (For IN, use comma as delimiter)
Table names	=	
Table creators	=	
Index names	=	
Index creators	=	
Cost in milliseconds	>	
Cost in service units	=	

Meet any filter condition  Meet all filter conditions

Retrieve Statements Cancel Help

## VE Static SQL Input – Access Path Filters

Static SQL Filtering - TUTORIAL

Filter name  
x

Only list explainable statements  List all static statements

New Save Save As ... Delete

Packages Plans Cost/object Filters **Access Path Filters** Summary

Use access path filter(Which requires that the plan\_table have the corresponding records)

Select the access path criteria for the static SQL statements that you want to list

<input checked="" type="checkbox"/> Sorts	<input checked="" type="checkbox"/> Non-matching index access	<input checked="" type="checkbox"/> Merge scan join
<input checked="" type="checkbox"/> Table space scans	<input checked="" type="checkbox"/> Matching index access	<input checked="" type="checkbox"/> Nested loop join
<input checked="" type="checkbox"/> List prefetch	<input checked="" type="checkbox"/> Non-index only access	<input checked="" type="checkbox"/> Hybrid join
<input checked="" type="checkbox"/> Sequential prefetch	<input checked="" type="checkbox"/> Multiple index access	<input checked="" type="checkbox"/> Full Outer Join
<input checked="" type="checkbox"/> CP parallelism	<input checked="" type="checkbox"/> In-list index access	<input checked="" type="checkbox"/> Left Outer Join
<input checked="" type="checkbox"/> I/O parallelism	<input checked="" type="checkbox"/> One-fetch access	<input checked="" type="checkbox"/> Star Join
<input checked="" type="checkbox"/> Sysplex query parallelism	<input checked="" type="checkbox"/> Index only access	<input checked="" type="checkbox"/> Inner Join
<input checked="" type="checkbox"/> Parallelism determined at run time		

Select all  
Clear all

Access Path Filters



## VE Dynamic Statement Cache Input

Visual Explain  
Subsystem Tools Plugin Properties Windows Help

List Databases

List cache statements-PATSV8EC

Statement Cache Input Table qualifier: SYSADM

Name	Operator	Value(For IN,use comma as delimiter)	Sort
Statement ID	=		No Sort
Statement Token	=		No Sort
Program Name	=		1st Asc
SQL text	=		1st Desc
Num of current users	=		3rd Asc
Num of copies	=		3rd Desc
Line Number	=		No Sort
Primary authorization ID	=		No Sort
Current SQLID	=		No Sort
Num of Executions	=		No Sort
Num of getpages	=		No Sort
Num of synchronous buffer reads	=		No Sort
Number of synchronous buffer write	=		No Sort
Number of rows examined	=		No Sort
Number of rows processed	=		No Sort
Number of sorts	=		No Sort
Number of index scans	=		No Sort
Number of tablespace scans	=		No Sort
Number of parallel groups	=		No Sort
Accumulated elapsed time	=		No Sort
Accumulated CPU time	=		No Sort
Synchronous I/O Wait Time	=		No Sort

Field Explanation:  
DSN\_STATEMENT\_CACHE\_TABLE.STMT\_ID  
**Statement ID:** Statement ID , Edm unique token

Ok Clear Cancel

## VE Dynamic Statement Cache Input

Visual Explain

Subsystem Tools Plugin Properties Windows Help

List cache statements-PATSV8EC

Statement View

All the rows are displayed. The number of the rows is 181.

STMT_ID	STMT_TOKEN	COLLID	PROGRAM_NAME	INV_DROPALT	INV_REVOKE	INV_LRU	INV_RUNSTATS	CA
10442		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-13 07:4
10337		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-13 07:1
9916		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-11 11:3
10308		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-12 12:4
10381		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-13 07:1
10354		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-13 07:1
10447		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-15 17:4
10403		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-13 07:1
10313		DSNDYNAMICSQLCACHE	SYSLH200	N	N	N	N	2005-01-12 12:5
10429		DSNDYNAMICSQLCA						7:1
1028		DSNDYNAMICSQLCA	10363			DSNDYNAMICSQLCACHE		8:0
1027		DSNDYNAMICSQLCA	10370			DSNDYNAMICSQLCACHE		8:0
10385		DSNDYNAMICSQLCA				DSNDYNAMICSQLCACHE		7:1
10368		DSNDYNAMICSQLCA	10349			DSNDYNAMICSQLCACHE		7:1
10370		DSNDYNAMICSQLCA				DSNDYNAMICSQLCACHE		7:1
10349		DSNDYNAMICSQLCA	10312			DSNDYNAMICSQLCACHE		7:1
10312		DSNDYNAMICSQLCA	102			DYNAMICSQLCACHE		2:5
102		DYNAMICSQLCA	1038			DYNAMICSQLCACHE		8:0
1038		DYNAMICSQLCA	103			DYNAMICSQLCACHE		7:1
103		DYNAMICSQLCA	103			DYNAMICSQLCACHE		2:4
10317		DSNDYNAMICSQLCA	103			DYNAMICSQLCACHE		3:1
10332		DSNDYNAMICSQLCA	10317			DSNDYNAMICSQLCACHE		7:1
10299		DSNDYNAMICSQLCA				DSNDYNAMICSQLCACHE		8:0
10362		DSNDYNAMICSQLCA	10332			DSNDYNAMICSQLCACHE		7:1
10289		DSNDYNAMICSQLCA				DSNDYNAMICSQLCACHE		8:0

Right - Click

Graph access path  
Show SQL text

Graph access path  
Show SQL text

## Using Visual Explain – Input/Explain Query

QueryNo: 2 SQLID: SYSADM8

Command history category: default

Name	SQL Statement	Comm
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 8:11:59 AM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 8:17:32 AM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 2:27:33 PM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 2:59:41 PM
Q1	SELECT EML_MES...	This query was saved at Dec 6, 2004 1:39:40 PM
Q1	SELECT CASE_ID ...	This query was saved at Dec 7, 2004 1:40:59 PM
Q1	SELECT CASE_ID ...	This query was saved at Dec 10, 2004 11:48:19 AM
Q1	SELECT * FROM ...	This query was saved at Dec 10, 2004 11:49:42 AM
Q1	SELECT * FROM ...	This query was saved at Dec 10, 2004 11:49:49 AM
Q1	SELECT DISTINCT ...	This query was saved at Dec 10, 2004 1:43:59 PM
Q1	SELECT DISTINCT ...	This query was saved at Dec 10, 2004 1:50:43 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:43:32 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:44:37 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:46:06 PM

Current degree: System default

Current refresh age: 0

Current maintained table types: None

Table qualifier for EXPLAIN stored procedure:

Input a SQL statement or select one from command history:

```
SELECT COUNT(*)
FROM
  PDB2.CONTACTNOTICE AS CN
  INNER JOIN TDB2.KEY_NOTICE AS KN
    ON CN.COD_GENERATION=KN.COD_GENERATION
  AND CN.ID_CONTACTNOTICE=KN.ID_CONTACTNOTICE
  INNER JOIN PDB2.CUST AS CU
    ON KN.COD_GENERATION=CU.COD_GENERATION
  AND KN.ID_CUST_JOB=CU.ID_CUST
```

Messages for Execution and Explain

**Input SQL Here**

**Click to Explain**

Explain with stored procedure Explain Execute





```

SELECT *
FROM PDB2.CONTACTNOTICE AS CN
  INNER JOIN TDB2.KEY_NOTICE AS KN
    ON CN.COD_CLIENT      =KN.COD_CLIENT
   AND CN.COD_GENERATION =KN.COD_GENERATION
   AND CN.ID_CONTACTNOTICE=KN.ID_CONTACTNOTICE
  INNER JOIN PDB2.CUST AS CU
    ON KN.COD_CLIENT      =CU.COD_CLIENT
   AND KN.COD_GENERATION =CU.COD_GENERATION
   AND KN.ID_CUST_JOB     =CU.ID_CUST
  INNER JOIN PDB2.CUST AS CU2
    ON KN.COD_CLIENT      =CU2.COD_CLIENT
   AND KN.COD_GENERATION =CU2.COD_GENERATION
   AND KN.ID_CUST_1       =CU2.ID_CUST
  INNER JOIN TDB2.KEY_PERSON AS KP
    ON CU2.COD_CLIENT     =KP.COD_CLIENT
   AND CU2.COD_GENERATION =KP.COD_GENERATION
   AND CU2.ID_CUST       =KP.ID_CUST
WHERE CN.DOM_NOTIFY      = 'FICH'
   AND CN.COD_GENERATION = 'MB'
   AND CN.COD_CLIENT     = '0450'
   AND CN.DOM_STATUS_NOTICE = 'ERL'
   AND CU2.DOM_PERSONGROUP = 'PR'
   AND CU2.COD_LAND_DIVISION = 'AT'
   AND CU2.DOM_STATUS     = 'BEST'
   AND CU2.DOM_CARE_STATE  = 'S';
    
```

## Query Example

- Query performs poorly
  - 20+ min, expectation 1 min
  - Need to determine why?
- Base RUNSTATS are current
  - Let's assume that at least the basics are covered
  - Not always a good assumption!!!

## Major causes of SQL performance problems

- Multi-table join
  - Poor choice of leading table
    - Insufficient statistics resulting in incorrect table chosen first
    - Indexes do not support most efficient table as leading
  - Inefficient join method or index usage on subsequent table(s)
    - Insufficient statistics resulting in poor estimate for current and/or prior tables accessed
    - Indexes do not support join (and possibly local) filtering
- Single table
  - Correct index or access method not chosen
    - Insufficient statistics to correctly decipher access choices
    - Indexes may not support filtering predicates

## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process

## Query Breakdown – Breaking apart the SQL

- Separate the query into a single count for each table
  - Applying local predicates to each

```
SELECT COUNT(*)
FROM   PDB2.CONTACTNOTICE AS CN
WHERE  CN.DOM_NOTIFY      = 'FICH'
       AND CN.COD_GENERATION = 'MB'
       AND CN.COD_CLIENT    = '0450'
       AND CN.DOM_STATUS_NOTICE = 'ERL'
```

```
SELECT COUNT(*)
FROM   TDB2.KEY_PERSON AS KP
```

```
SELECT COUNT(*)
FROM   TDB2.KEY_NOTICE AS KN
```

```
SELECT COUNT(*)
FROM   PDB2.CUST AS CU
```

```
SELECT COUNT(*)
FROM   PDB2.CUST AS CU2
WHERE  CU2.DOM_PERSONGROUP = 'PR'
       AND CU2.COD_LAND_DIVISION = 'AT'
       AND CU2.DOM_STATUS      = 'BEST'
       AND CU2.DOM_CARE_STATE   = 'S';
```

**Is that all the local predicates?**



## Query Breakdown – Applying all local predicates

- Consider transitively closed predicates also
  - Be aware of restrictions such as LIKE, IN, subqueries and expressions
  - Based upon join and local predicates
    - CN.COD\_GENERATION=KN.COD\_GENERATION
    - CN.COD\_CLIENT =KN.COD\_CLIENT
    - KN.COD\_GENERATION=CU.COD\_GENERATION
    - KN.COD\_CLIENT =CU.COD\_CLIENT
    - KN.COD\_GENERATION=CU2.COD\_GENERATION
    - KN.COD\_CLIENT =CU2.COD\_CLIENT
    - CU2.COD\_GENERATION=KP.COD\_GENERATION
    - CU2.COD\_CLIENT =KP.COD\_CLIENT
    - CN.COD\_GENERATION = 'MB' } Also apply to KN,
    - CN.COD\_CLIENT = '0450' } CU, CU2, KP

## Query Breakdown – Counts

```
SELECT COUNT(*) = 1,472
FROM PDB2.CONTACTNOTICE AS CN
WHERE CN.DOM_NOTIFY          = 'FICH'
      AND CN.COD_GENERATION   = 'MB'
      AND CN.COD_CLIENT       = '0450'
      AND CN.DOM_STATUS_NOTICE = 'ERL'
```

```
SELECT COUNT(*) = 20,114
FROM TDB2.KEY_PERSON AS KP
WHERE KP.COD_CLIENT          = '0450'
      AND KP.COD_GENERATION  = 'MB'
```

```
SELECT COUNT(*) = 156,347
FROM TDB2.KEY_NOTICE AS KN
WHERE KN.COD_CLIENT          = '0450'
      AND KN.COD_GENERATION  = 'MB'
```

```
SELECT COUNT(*) = 420,973
FROM PDB2.CUST AS CU
WHERE CU.COD_CLIENT          = '0450'
      AND CU.COD_GENERATION  = 'MB'
```

```
SELECT COUNT(*) = 267,011
FROM PDB2.CUST AS CU2
WHERE CU2.DOM_PERSONGROUP   = 'PR'
      AND CU2.COD_LAND_DIVISION = 'AT'
      AND CU2.DOM_STATUS     = 'BEST'
      AND CU2.DOM_CARE_STATE  = 'S'
      AND CU2.COD_CLIENT      = '0450'
      AND CU2.COD_GENERATION  = 'MB'
```

**\*\* Generally want most  
filtered table accessed first**

## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process

## Using Visual Explain - Access Path Analysis

SQL Text | Access Plan | Execution Result | Report | Plan Hint | Statistics Advisor

DB2 Platform: Z/OS DB2 Version: v8 Explain Time: 2005-02-02 11:02:06.42

table(CUST)

- Columns
- Indexes
- Tablespace
- Table Partitions

Show attribute explanation Views: Cost estimation

Name	Value
Name	CUST
Creator	PDB2
Correlation Name	CU2
Qualifying Rows	1
Rows	2.8926293E7
Pages	2041274
Compressed Row Percentage	99
Timestamp	2004-06-09 10:38:14
Explain Time	2005-02-02 11:02:06

Attribute explanation:  
Name: Table name

**Click on object to obtain cost info**

**1<sup>st</sup> table accessed - CUST**



## Using Visual Explain – Cost Estimates

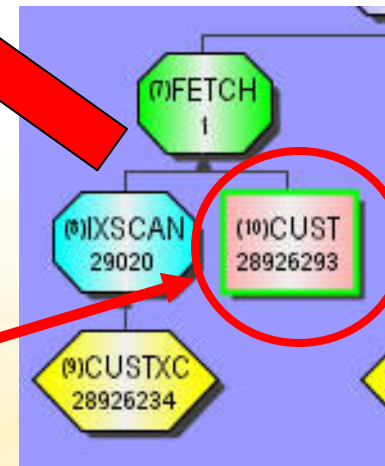
Show attribute explanation Views: Cost estimation

Name	Value
Name	CUST
Creator	PDB2
Correlation Name	CU2
Qualifying Rows	1
Rows	2.8926293E7
Pages	2041274
Compressed Row Percentage	99
Timestamp	2004-06-09 10:38:14
Explain Time	2005-02-02 13:37:51

CUST table (CU2)

Estimate 1 row qualifies

Click on table object for statistics



## Using Visual Explain - Cost Estimates



Click on Report  
Tab for summary

**Access Path Report**

[Query Summary](#)

---

**[Table Summary](#)**

---

[Predicate Summary](#)

### Table Summary

Table Creator	Table Name	Correlation Name	Rows	Pages	Qualified Rows
PDB2	CONTACTNOTICE	CN	7.316933E7	3123077	704.0725
PDB2	CUST	CU2	2.8926293E7	2041274	1
TDB2	KEY_NOTICE	KN	6.1395078E7	804496	156163.44
PDB2	CUST	CU	2.8926293E7	2041274	58039.527
TDB2	KEY_PERSON	KP	2.4216504E7	470785	19619.812

Choose Table Summary (Predicate Summary used in later step)

## Comparing Table Counts Vs Cost Estimates

- How do the counts compare with estimates?

Reason CU2  
accessed 1st

Table	Count	% of cardf	Estimate	% of cardf
CUST (CU)	420,973	1.45%	58,039	0.2%
CUST (CU2)	267,011	0.92%	1	0.000003%
CONTACTNOTICE	1,472	0.002%	704	0.0009%
KEY_PERSON	20,114	0.03%	19,619	0.03%
KEY_NOTICE	156,347	0.65%	156,163	0.65%

From COUNTs

From Table Summary

## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process





## What do I do if the estimate is incorrect?

- Start by narrowing your scope.....
  - What estimate is incorrect?
    - “Qualified row estimates” vs “Real table counts” are incorrect
  - But there is more than one that is incorrect?
    - Focus on the worst one – CUST (CU2)
  - Where to next?
    - Table qualified row estimates are a combination of individual predicate estimates.....**so drill down further to the predicates.**

## Predicate Report

### Predicate Summary

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
2	COD_GENERATION	6	EQUAL	VALUE		0.1695
3	DOM_PERSONGROUP	18	EQUAL	VALUE		0.0556
4	COD_LAND_DIVISION	461	EQUAL	VALUE		0.0022
5	DOM_STATUS	57	EQUAL	VALUE		0.0175
6	DOM_CARE_STATE	24	EQUAL	VALUE		0.0417
7	COD_CLIENT	169	EQUAL	VALUE		0.0059

How does this match reality? Focus on 1 predicate at a time

## Comparing predicate estimate with reality

- CU2.COD\_LAND\_DIVISION = 'AT'

- Run count

```
SELECT COUNT(*) = 26,149,368
FROM PDB2.CUST AS CU2
WHERE CU2.COD_LAND_DIVISION = 'AT'
```

Only 99.7%  
wrong!!!

Count	Cardf	Count / cardf	Filter factor
26,149,368	28,926,293	0.904 (90.4%)	0.0022 (0.22%)

- How can actual percentage (from count) differ from filter factor?
  - Data must not be evenly distributed (data is skewed)

## Evaluating Data Skew

- Run the query below
  - Result clearly shows that data is not evenly distributed

```
SELECT COD_LAND_DIVISION, COUNT(*)
FROM PDB2.CUST
GROUP BY COD_LAND_DIVISION
ORDER BY 2 DESC
```



COD_LAND_DIVISION		
-----+-----+-----		
AT	26149368	(90.4%)
	960390	
DE	841482	
CZ	173971	
HU	117924	
SI	92994	
IT	92763	
... Not all displayed		

## Comparing Predicate Counts Vs Cost Estimates

- First estimate is correct, others are disasters

Predicate	Count	Filter Factor Estimate
COD_GENERATION = 'MB'	0.1695	0.1695 ✓
DOM_PERSONGROUP = 'PR'	0.8643	0.0556 ✗
COD_LAND_DIVISION = 'AT'	0.9041	0.0022 ✗
DOM_STATUS = 'BEST'	0.9575	0.0175 ✗
DOM_CARE_STATE = 'S'	0.8206	0.0417 ✗
COD_CLIENT = '0450'	0.08	0.0059 ✗

↑  
Calculated as count / table cardf  
(note 0.1695 = 16.95%)



## RUNSTATS to collect Data Skew

- Run RUNSTATS on CUST table
  - V8 COLGROUP keyword allows frequencies on non-indexed columns

**RUNSTATS TABLESPACE IDVKUNDA.IDVCUST**

**TABLE(PDB2.CUST)**

**COLGROUP(COD\_CLIENT)                      FREQVAL COUNT 10**

**COLGROUP(DOM\_STATUS)                     FREQVAL COUNT 10**

**COLGROUP(COD\_LAND\_DIVISION) FREQVAL COUNT 10**

**COLGROUP(DOM\_PERSONGROUP) FREQVAL COUNT 10**

**COLGROUP(DOM\_CARE\_STATE)               FREQVAL COUNT 10**

## New Predicate Report

### Predicate Summary

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
2	COD_GENERATION	6	EQUAL	VALUE		0.1695
3	DOM_PERSONGROUP	18	EQUAL	VALUE		0.8643
4	COD_LAND_DIVISION	461	EQUAL	VALUE		0.9041
5	DOM_STATUS	57	EQUAL	VALUE		0.9575
6	DOM_CARE_STATE	24	EQUAL	VALUE		0.8206
7	COD_CLIENT	169	EQUAL	VALUE		0.08

Count
<b>0.1695</b>
<b>0.8643</b>
<b>0.9041</b>
<b>0.9575</b>
<b>0.8206</b>
<b>0.08</b>



How well does this match reality? Perfectly now

**\*\* But don't always expect perfection \*\***

## New Table Report

### Table Summary

Table Creator	Table Name	Correlation Name	Rows	Pages	Qualified Rows	Count
PDB2	CUST	CU2	2.8926293E7	2041274	240891.62	<b>267,011</b>
TDB2	KEY_PERSON	KP	2.4216504E7	470785	19619.812	<b>20,114</b>
PDB2	CUST	CU	2.8926293E7	2041274	419204.44	<b>420,973</b>
TDB2	KEY_NOTICE	KN	6.1395078E7	804496	156163.44	<b>156,347</b>
PDB2	CONTACTNOTICE	CN	7.316933E7	3123077	704.0725	<b>1,472</b>

How well does this match reality? Not Perfect, but closer



## Revised Access Path

- Query now performs well
  - < 20 sec
  - original > 20 min

SQL Text | Access Plan | Execution Result | Report | Plan Hint | Statistics Advisor

DB2 Platform: Z/OS DB2 Version: v8 Explain Time: 2005-02-04 10:11:38.36

table(CONTACTNOTICE)

- Columns
- Indexes
- Tablespace
- Table Partitions

Show attribute explanation Views: Cost estimation

Name	Va
Name	CONTACTNOTICE
Creator	PDB2
Correlation Name	CN
Qualifying Rows	704.0725
Rows	7.316933E7
Pages	3123077
Compressed Row Percentage	99
Timestamp	2004-06-08 04:09:31
Explain Time	2005-02-04 10:11:38

Attribute explanation:  
Name: Table name

## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process



## Automating the SQL Tuning Process

The screenshot shows the 'Tune SQL - STLEC1' application window. The interface includes a menu bar (File, Edit, Action, View), a toolbar with icons for file operations and SQL execution, and a main workspace divided into several sections.

**Command History Section:**

QueryNo: 3      SQLID: SYSADM8

Command history category: default

Name	SQL Statement	Comm
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 8:11:59 AM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 8:17:32 AM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 2:27:33 PM
Q1	SELECT "9AMATN...	This query was saved at Dec 2, 2004 2:59:41 PM
Q1	SELECT EML_MES...	This query was saved at Dec 6, 2004 1:39:40 PM
Q1	SELECT CASE_ID ...	This query was saved at Dec 7, 2004 1:40:59 PM
Q1	SELECT CASE_ID ...	This query was saved at Dec 10, 2004 11:48:19 AM
Q1	SELECT * FROM ...	This query was saved at Dec 10, 2004 11:49:42 AM
Q1	SELECT * FROM ...	This query was saved at Dec 10, 2004 11:49:49 AM
Q1	SELECT DISTINCT ...	This query was saved at Dec 10, 2004 1:43:59 PM
Q1	SELECT DISTINCT ...	This query was saved at Dec 10, 2004 1:50:43 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:43:32 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:44:37 PM
Q1	SELECT * FROM ...	This query was saved at Dec 13, 2004 3:46:06 PM

Buttons: Create New Category, Rename Category, Delete Category, Up, Down, Update Name, Update Statement, Update Comment, Change Category, Copy to Category, Delete Statement.

Automatically save SQL in current command history category

**Right Panel:**

Current degree: System default  
 Current refresh age: 0  
 Current maintained table types: None  
 Table qualifier for EXPLAIN stored procedure:   
 Input a SQL statement or select one from command history:

```
SELECT COUNT(*)
FROM PDB2.CONTACTNOTICE AS CN INNER JOIN TDB2.KEY_NOTICE
AS KN
ON CN.COD_GENERATION=KN.COD_GENERATION
AND CN.ID_CONTACTNOTICE=KN.ID_CONTACTNOTICE
INNER JOIN PDB2.CUST AS CU
ON KN.COD_GENERATION=CU.COD_GENERATION
AND KN.ID_CUST_JOB=CU.ID_CUST
INNER JOIN PDB2.CUST AS CU2
```

Messages for Execution and Explain

The above query has been explained successfully

**Closeup View**

Analyze

Buttons: Explain with stored procedure, Explain, Execute, Plan Hint, Analyze, Help.

Tune SQL - STLEC1  
View Statistics Advisor

SQL Text | Access Plan | Execution Result | Report | Plan Hint | Statistics Advisor |  
Runstats | Explanation | Conflict Report |

```
RUNSTATS TABLESPACE IDVKUNDA.IDVKONTM
TABLE(PDB2.CONTACTNOTICE) SAMPLE 5
COLUMN(ID_CONTACTNOTICE,COD_CLIENT,DOM_NOTIFY,DOM_STATUS_NOTICE)
COLGROUP(COD_CLIENT) FREQVAL COUNT 10
COLGROUP(DOM_NOTIFY) FREQVAL COUNT 10
COLGROUP(DOM_STATUS_NOTICE) FREQVAL COUNT 10
SORTDEVT SYSDA
INDEX(IDVIX.CONTACTNOTICEX1,
      IDVIX.CONTACTNOTICEX3,
      IDVIX.CONTACTNOTICEX2 KEYCARD,
      IDVIX.CONTACTNOTICEX4,
      IDVIX.CONTACTNOTICEXC KEYCARD,
      IDVIX.CONTACTNOTICEX5 KEYCARD)
SHRLEVEL CHANGE REPORT YES

RUNSTATS TABLESPACE IDVKUNDA.IDVCUST
TABLE(PDB2.CUST) SAMPLE 5
COLUMN(ID_CUST,DOM_PERSONGROUP,COD_CLIENT,DOM_STATUS,DOM_CARE_STATE,
      COD_LAND_DIVISION)
COLGROUP(COD_CLIENT,COD_GENERATION,COD_LAND_DIVISION,DOM_CARE_STATE,DOM_PERSON
COLGROUP(DOM_PERSONGROUP) FREQVAL COUNT 10
COLGROUP(COD_CLIENT) FREQVAL COUNT 10
COLGROUP(DOM_STATUS) FREQVAL COUNT 10
COLGROUP(DOM_CARE_STATE) FREQVAL COUNT 10
COLGROUP(COD_CLIENT,COD_GENERATION,ID_CUST)
COLGROUP(COD_LAND_DIVISION) FREQVAL COUNT 10
SORTDEVT SYSDA
INDEX(IDVIX.CUSTX7,
      IDVIX.CUSTX3,
      IDVIX.CUSTX5 KEYCARD,
```

Save As | Execute RUNSTATS...

## Statistics Advisor – RUNSTATS Recommendations

Closeup View

Execute RUNSTATS...



# RUNSTATS Recommendations - Closeup

```

RUNSTATS TABLESPACE IDVKUNDA.IDVCUST
TABLE(PDB2.CUST) SAMPLE 5
COLUMN(ID_CUST, DOM_PERSONGROUP, COD_CLIENT, DOM_STATUS
, DOM_CARE_STATE, COD_LAND_DIVISION)
COLGROUP(COD_CLIENT, COD_GENERATION, COD_LAND_DIVISION
, DOM_CARE_STATE, DOM_PERSONGROUP, DOM_STATUS)
COLGROUP(COD_CLIENT, COD_GENERATION, ID_CUST)
COLGROUP(DOM_PERSONGROUP) FREQVAL COUNT 10
COLGROUP(COD_CLIENT) FREQVAL COUNT 10
COLGROUP(DOM_STATUS) FREQVAL COUNT 10
COLGROUP(DOM_CARE_STATE) FREQVAL COUNT 10
COLGROUP(COD_LAND_DIVISION) FREQVAL COUNT 10
    
```

Column  
Statistics

Correlation  
Statistics

Frequency Statistics  
– Same as manual evaluation

## Statistics Advisor

- Automated statistics determination
  - Often queries have inefficient OR unstable performance due to lack of statistics
  - SA automates the analysis of statistics required for an SQL statement
- Goal
  - Automate SOLUTION to many common SQL performance problems
  - Solve SQL performance problems quickly and easily



## Statistics Advisor or Manual Analysis?

- Statistics Advisor is a 1<sup>st</sup> step
  - May resolve the majority of queries with unstable or inefficient access paths
  - Although currently only single query based, RUNSTATS recommendations improve optimizer's knowledge for all queries
- Deeper manual analysis may still be required
  - You may wish to validate the recommendations from SA
  - SA makes assumptions about need for frequency or correlation statistics, run counts to verify real need.
- Other problems may still exist
  - Inadequate indexing, inefficient predicates etc.



## Agenda

- Simple methods to find the problem query
- Query breakdown
- Comparing estimates with reality
- What do I do if the estimate is incorrect?
- Automating the process
- **What if this presentation didn't cover my SQL problem?**

## Try Another Problem

- Table count does not match estimate

```

SELECT COUNT(*) = 114,856
FROM SAPR3.PAYR
WHERE REGION = 'K03'
      AND DIV   = 'WFB2'
      AND DEPT  = 'ARPS'
    
```

114,856 vs 143

### Table Summary

Table Creator	Table Name	Correlation Name	Rows	Pages	Qualified Rows
SAPR3	PAYR		1.7267799E7	862780	143.2536

## Predicate Counts

- Run Predicate counts

### Predicate Summary

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
2	REGION	35	EQUAL	VALUE		0.0286
3	DIV	42	EQUAL	VALUE		0.0238
4	DEPT	82	EQUAL	VALUE		0.0122

**SELECT COUNT(\*) = 302,949**  
**FROM SAPR3.PAYR**  
**WHERE DEPT = 'ARPS'**

**SELECT COUNT(\*) = 314,174**  
**FROM SAPR3.PAYR**  
**WHERE REGION = 'K03'**

**SELECT COUNT(\*) = 302,949**  
**FROM SAPR3.PAYR**  
**WHERE DIV = 'WFB2'**

## Predicates - Actual vs Estimates

- Compare Actual vs Estimate

Predicate	Count	Cardf	Count / cardf	Filter Factor
REGION = 'K03'	314,174	17,267,799	0.0182	0.0286
DIV = 'WFB2'	302,949	17,267,799	0.0175	0.0238
DEPT = 'ARPS'	302,949	17,267,799	0.0175	0.0122

Filter Factor – Actual vs Estimate - Not Perfect, but close  
So why the difference in table actual vs estimate?

## Detecting Correlation - Counts

- Run Predicate counts
  - Distinct occurrences of each column

```
SELECT COUNT(DISTINCT REGION) = 35 REGIONs  
      ,COUNT(DISTINCT DIV)   = 42 DIVs  
      ,COUNT(DISTINCT DEPT)  = 82 DEPTs  
FROM SAPR3.PAYR
```

- Distinct occurrences of the column group

```
SELECT COUNT(*) = 167 Combinations of REGION, DIV, DEPT  
FROM  
(SELECT DISTINCT REGION, DIV, DEPT  
FROM SAPR3.PAYR) AS A
```



## Detecting Correlation - Calculation

- Calculation to detect correlation
  - If the product of the individual counts > group count
    - Then columns are correlated
      - Product of counts =  $35 * 42 * 82 = 120,540$
      - Group count = 167
      - $120,540 > 167$
    - Therefore, columns are correlated
- Trivia
  - Optimizer treats columns as independent unless statistics demonstrate otherwise
  - $17,267,799 * 1/35 * 1/42 * 1/82 = 143.2536$  ← Look familiar?

# Statistics Advisor Recommendations

```
SQL Text | Access Plan | Execution Result | Report | Plan Hint | Statistics Advisor |
Runstats | Explanation | Conflict Report |
RUNSTATS TABLESPACE A130#130.PAYR
          TABLE(SAPR3.PAYR) SAMPLE 5
          COLGROUP (DEPT,DIV,REGION)
          SORTDEVT SYSDA
SHRLEVEL CHANGE REPORT YES
```

COLGROUP used to collect correlation

## Revised Table Estimate

- Qualified Rows Estimate after RUNSTATS

Table Summary					
Table Creator	Table Name	Correlation Name	Rows	Pages	Qualified Rows
SAPR3	PAYR		1.7267799E7	862780	103399.94

- Why do we care?**
  - For single table access
    - May influence choice of access method, including index choice, usage of list or sequential prefetch etc
  - For multi-table access
    - Qualified row estimate used as input to subsequent tables. Choice of join sequence, join method, and access method (index?) for each table.

Closer to 114,856 (count)

## Summary

- Many more examples possible
- Basic fundamentals remain the same
  - Run counts to compare table/predicate estimates with reality to:
    - Detect correlation,
    - Identify data skew,
    - Find optimistic or poor filter factors (eg. Range predicate with parameter markers or column expressions).
- Using Statistics Advisor
  - So simple that even the help desk could do it!!
- Using the method outlined in this presentation
  - Take query tuning to the next level and tune SQL like an expert

*Tune SQL like an expert in DB2 UDB for z/OS V8*  
*Session: G7*

## Thankyou for listening!!!

**Terry Purcell**

IBM Silicon Valley Lab

*tpurcel@us.ibm.com*