IDUG

2026

Sydney | March 16 - 18

AU Db2 TECH CONFERENCE

Db2 for z/OS Tuning Techniques: Practical Strategies for Maximum Performance

Devanand Karunakaran, BMC Software

Session Code: A04

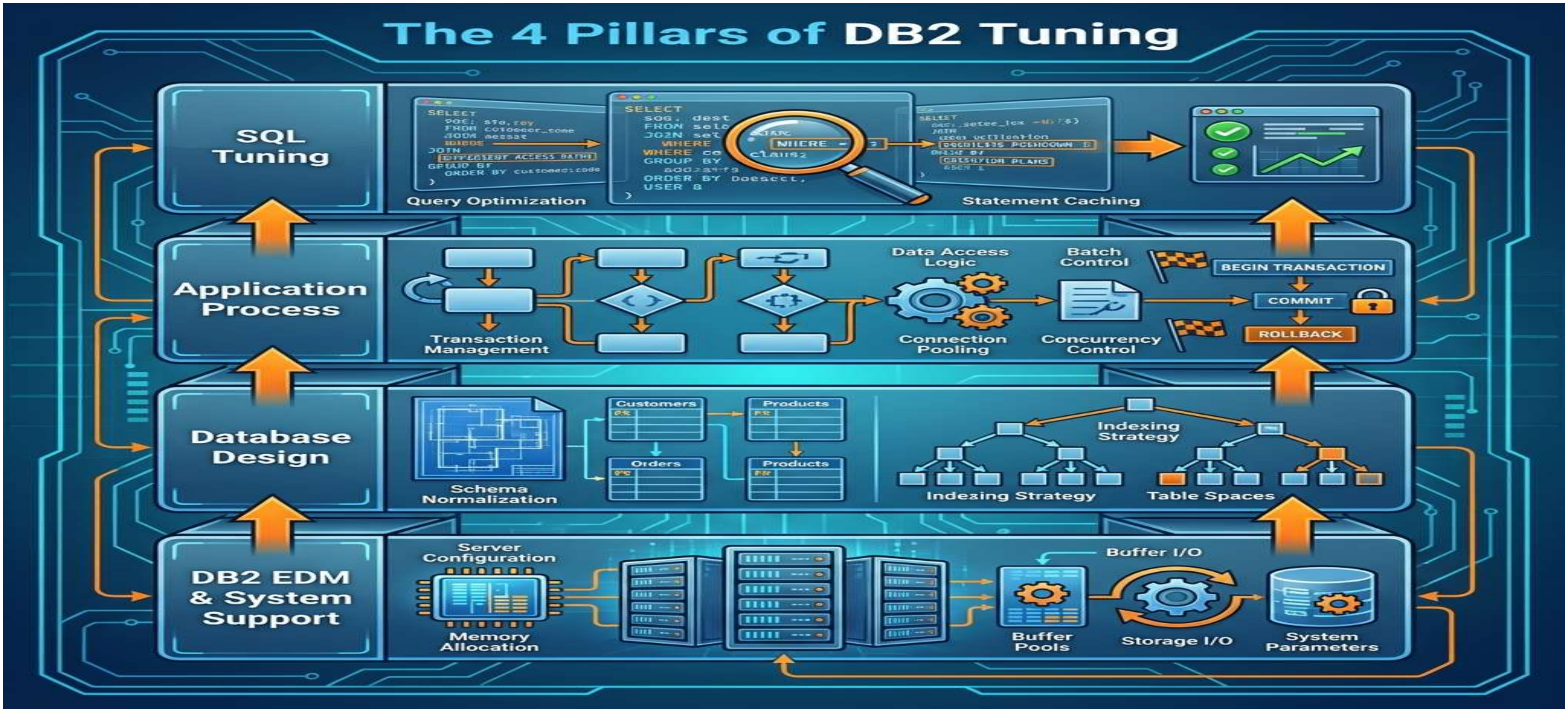
Platform:

Db2 for z/OS

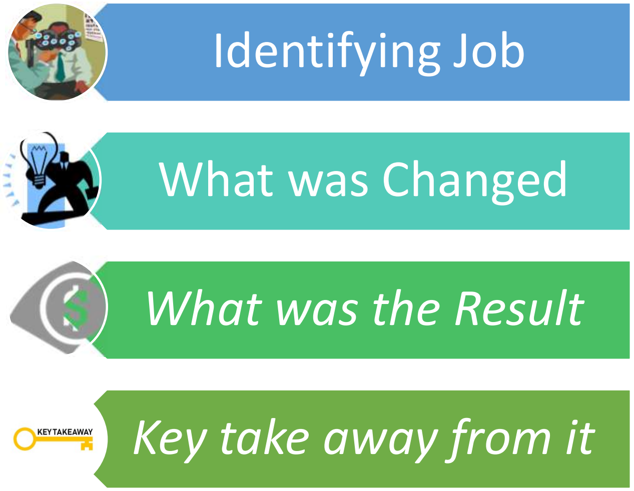
AGENDA

- Different Db2 Tuning Techniques –Overview
- Realtime Tuning examples
    - ❖SQL Tuning
    - ❖Application Process Change
    - ❖Database Design Change
    - ❖EDM Pool Tuning

Db2 Tuning Techniques



DB2 Performance Tuning



Identifying Job

What was Changed

What was the Result

Key take away from it

IDUG
2026 Australia Db2 Tech Conference

SQL Tuning - SQL Rewrite (1/4)

#IDUGDb2

- Application job that runs 5days a week and consumes an average of 370 CPU Sec per day. **CPU aggressive - Over 100 CPUSec/Min**

- Had 2 SQL s with date validation, DB2 Timestamp Column CREATION_DATE was converted to Date and used in the predicate.

```
WHERE DATE(CREATION_DATE) = (CURRENT DATE - 1 DAY)

WHERE DATE(CREATION_DATE) BETWEEN (CURRENT DATE - 3 DAYS)
AND (CURRENT DATE - 1 DAY)
```

- Even though there was an index on the DB2 Timestamp field CREATION_DATE, it was not used as part of the access because of the date conversion.

SQL Tuning - SQL Rewrite (2/4)

- **Changed the SQL to make the job use the existing index.**

```
WHERE DATE(CREATION_DATE) = (CURRENT DATE - 1 DAY)
```

Changed To

```
WHERE CREATION_DATE  BETWEEN TIMESTAMP(CHAR(CURRENT DATE- 1 DAY),'00:00:00')
AND TIMESTAMP(CHAR(CURRENT DATE- 1 DAY),‘24:00:00')
```

```
WHERE DATE(CREATION_DATE) BETWEEN (CURRENT DATE - 3 DAYS)
AND  (CURRENT DATE - 1 DAY)
```

Changed To

```
WHERE CREATION_DATE BETWEEN TIMESTAMP(CHAR(CURRENT DATE- 3 DAYS),'00:00:00')

AND TIMESTAMP(CHAR(CURRENT DATE- 1 DAY),'00:00:00')
```

SQL Tuning - SQL Rewrite (3/4)

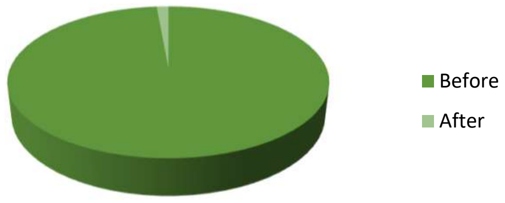
Before Fix

JOBNAME-	JOBID-	RUNON	DAY	ELAPS	CPUSEC
JOB0001	J22147	NOV07	THU	0:03	340.6
JOB0001	J10312	NOV06	WED	0:03	347.2
JOB0001	J30422	NOV05	TUE	0:04	457.6
JOB0001	J32611	NOV02	SAT	0:04	341.9
JOB0001	J08629	NOV01	FRI	0:03	342.3
			Weekly CPU Usage -		**1829.6**

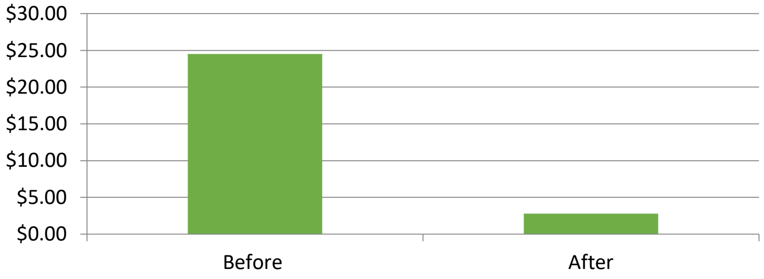
After Fix

JOBNAME-	JOBID-	RUNON	DAY	ELAPS	CPUSEC
JOB0001	J28727	NOV16	SAT	0:00	7.1
JOB0001	J32480	NOV15	FRI	0:01	4.9

Daily CPU Usage of the Job



Daily Job Cost



Minimum Per Annum saving will be AUD$6,000

SQL Tuning - SQL Rewrite (4/4)

KEY TAKEAWAY

- ✓ **Always code Stage 1 (Indexable) predicates**
- ✓ **Avoid Scalar Functions on Columns**
- ✓ **Select Only needed Columns – Index Only**
- ✓ **A job that consumes a high % of CPU/Sec could be a candidate for tuning.**
- ✓ **CPU aggressive jobs like 100CPUSec/Min – Most likely doing index scan or tablespace scan.**

SQL Tuning- Access path fix (1/4)

- Identified the issue with the highly used online Package
- The SQL processing of 3 tables, believed that changing the sequence of the table processing will yield optimal access to the data page.
- Tried a few combinations and picked the optimal access path.
- Forced DB2 to use the suggested access path through OPTHINT

SQL Tuning- Access path fix (2/4)

DB2 optimizer picked access path

```
PLANNO METHOD CREATOR          TNAME          TABNO  ACCESSTYPE MATCHCOLS
------ ------ -------- ------------------ ------ ---------- ---------
********************************** TOP ********************************
     1      0 CRETR1   TABLE1                 3     I             1   INDEX1A
     2      1 CRETR1   TABLE2                 1     I             2   INDEX2E
     3      1 CRETR1   TABLE3                 2     I             6   INDEX3A
+                        SQL Summary Information
+ ----------------------------------------------------------------------------
+ Sql Call         Stmt# Count   InDB2 Time   InDB2 CPU  Avg Time     Avg CPU
+ ---------------- ----- ------  ------------ ---------  ------------ ---------
+ PREPARE           1834      2  00:00.02727     .00160  00:00.01363     .00080
+ OPEN CURSOR       1934      1  00:00.00002     .00002  00:00.00002     .00002
+ FETCH             1964      2  00:00.06312     .05590  00:00.03156     .02795 --- Not good
+ CLOSE CURSOR      2238      1  00:00.00001     .00001  00:00.00001     .00001
```

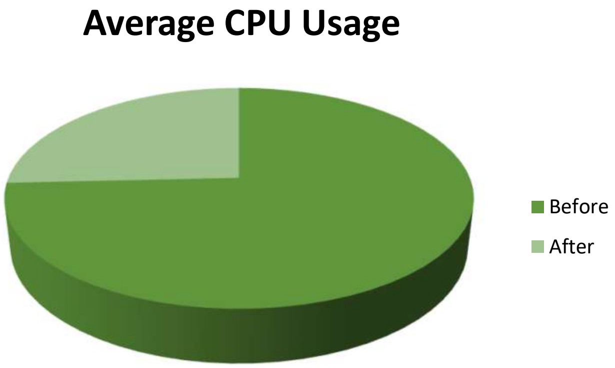
Opthint – Forced Access

```
 PLANNO METHOD CREATOR          TNAME          TABNO  ACCESSTYPE MATCHCOLS
------ ------ -------- ------------------ ------ ---------- ---------
********************************** TOP ********************************
     1      0 CRETR1   TABLE3                 2     I             5   INDEX3C
     2      1 CRETR1   TABLE2                 1     I             3   INDEX2A
     3      1 CRETR1   TABLE1                 3     I             1   INDEX1A
+                        SQL Summary Information
+ ----------------------------------------------------------------------------
+ Sql Call         Stmt# Count   InDB2 Time   InDB2 CPU  Avg Time     Avg CPU
+ ---------------- ----- ------  ------------ ---------  ------------ ---------
+ PREPARE           1834      3  00:00.00246     .00165  00:00.00082     .00055
+ OPEN CURSOR       1934      1  00:00.00001     .00001  00:00.00001     .00001
+ FETCH             1964      2  00:00.00028     .00027  00:00.00014     .00013 -Much Efficient
+ CLOSE CURSOR      2238      1  00:00.00001     .00001  00:00.00001     .00001
```

SQL Tuning- Access path fix (3/4)

There is around 71% improvement in the query performance after we rebinded PACKAGE1 to fix the access path with OPTHINT.

```
                                    Nbr of    --CPU Time--        --Svc Time--
Seqno  Name      Stmt# SQL Function SQL Calls  Total    Mean       Total    Mean
S00111 PACKAGE1   2962 SELECT           1,530   3.52    0.00230    23.08    0.01508 - Before Fix
S00178 PACKAGE1   2962 SELECT           1,932   1.22    0.00063    28.13    0.01456 - After FIX
```


Average CPU Usage

Minimum Per Annum saving will be AUD$20,000

SQL Tuning- Access path fix (4/4)

KEY TAKEAWAY

✓ Filter strict predicate early – to decide on table sequence

✓ Identify easily tunable SQL by looking at Average CPU usage and Get pages
(< 1 microsec CPU Time and less than 1000 Get pages)

✓ Jion Strategies – Nested loop for small sets and Merge Scan for sorted /large subset

General Considerations for SQL Tuning

❖ Avoid Tablespace scan for large tables

❖ Avoid sorting when possible

❖ Leverage the power of SQL to simplify processing

❖ Have a proper join predicate when multiple table data need to be processed

❖ Check CPU usage of dynamic SQL after the Db2 version upgrade

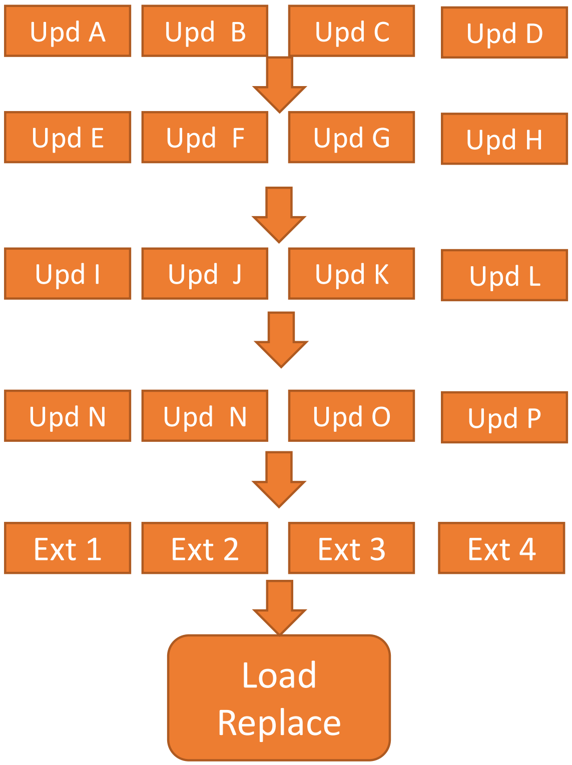
Application Process (1/5)

Complex Update Process – Tuning

- Daily Update a Large Partition table based on Criteria matching on four other tables.
- Business gives conditions to update the Large PBR table, and many times provides data to be used as a base for updating the records.
- 16 Updates Jobs – 4 Jobs run in parallel – Processing 64 partitions – 250 Million records in total - **Only less than 0.01% rows updated**.
- 4 Extract Jobs -Processing 64 partitions – 250 Millions records
- Load REPLACE Jobs every time to move updated records to the report extract table.

Application Process (2/5)

Re-Designed the Flow



Upd A Upd B Upd C Upd D

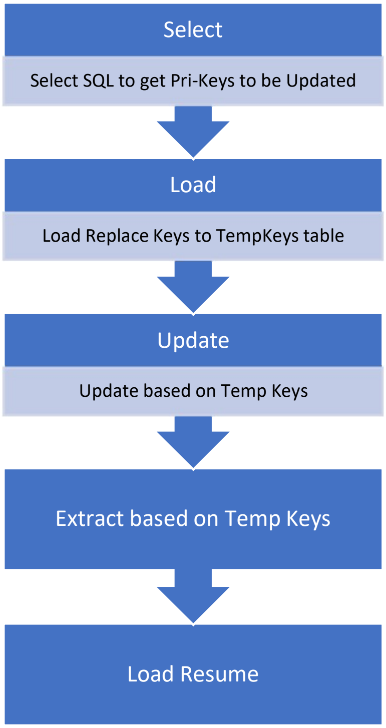
Upd E Upd F Upd G Upd H

Upd I Upd J Upd K Upd L

Upd N Upd N Upd O Upd P

Ext 1 Ext 2 Ext 3 Ext 4

Load Replace



Select

Select SQL to get Pri-Keys to be Updated

Load

Load Replace Keys to TempKeys table

Update

Update based on Temp Keys

Extract based on Temp Keys

Load Resume

Application Process (3/5)

Simplified Update Process

Re-Designed the Flow

- ➢ **Run Select SQL to get the record Keys to be updated**
- ➢ **Load REPLACE record keys to a Temp table**
- ➢ **Run Update based on the Keys**
- ➢ **Run Extract to extract held records for that day**
- ➢ **Load Resume to the report extract table**

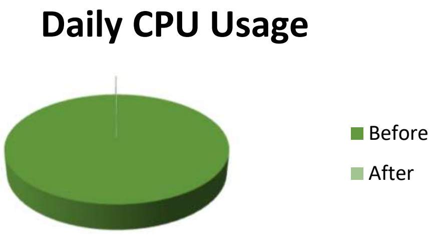
Application Process (4/5)

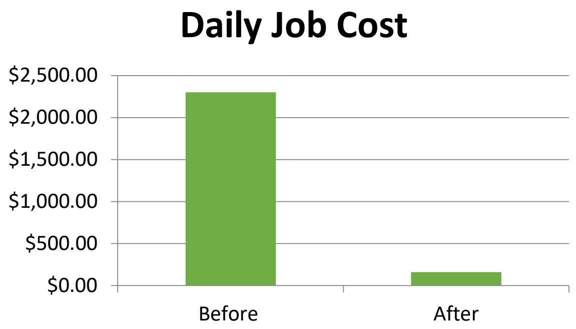
BEFORE

JOBNAME-	JOBID-	ELAPS	CPUSEC
JOBF1LDR	J25836	0:02	24.10
JOBF1EX1	J24833	0:48	177.20
JOBF1EX2	J24830	0:47	170.00
JOBF1EX3	J24831	0:47	166.90
JOBF1EX4	J24832	0:45	157.30
JOBF1014	J24428	0:14	520.10
JOBF1013	J23973	0:29	678.40
JOBF1015	J23959	0:26	553.50
JOBF1016	J24403	0:00	0.10
JOBF1012	J23804	0:37	762.40
JOBF1HLO	J24236	0:11	214.70
JOBF1011	J23535	0:37	748.80
JOBF1009	J23450	0:26	584.20
JOBF1007	J23351	0:33	779.60
JOBF1009	J23057	0:34	863.20
JOBF1008	J23043	0:23	615.20
JOBF1005	J22704	0:31	738.50
JOBF1002	J22111	0:52	1034.40
JOBF1004	J22113	0:38	667.70
JOBF1003	J22112	0:36	647.10
JOBF1001	J22110	0:26	477.30
ELAPS TIME		**642 Mins**	
Daily CPU Usage			**10580.70**

AFTER

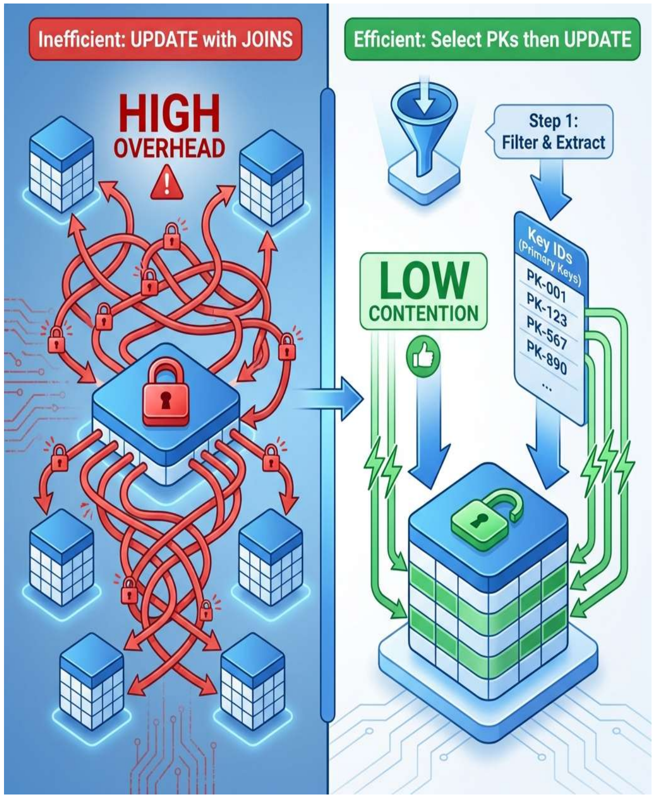
```
JOBNAME- --STEP-- -PSTEP-- CCODE ELAPSED-TIME -CPU-TIME-
JOB00001  STEP1    SELSTP      0     00:04:30      8.62S
JOB00001 LOADSTP1              0     00:00:04      0.47S
JOB00001 UPDSTP                0     00:01:19      2.77S
JOB00001 UNLDRPT               0     00:00:04      1.13S
JOB00001 LOADSTP2              4     00:00:14      1.30S

ELAPS TIME                          5.71 Mins

Daily CPU Usage                                   14.29S
```





Minimum Per Annum saving will be AUD$70,000

Application Process (5/5)



- UPDATE SQL have many limitations –Compared to SELECT
- SELECT is more efficient on complex SQL's
- UPDATE will be faster as it is based on the primary key.
- Back-out will be similar and easy.
- Check the final output record counts and compare with the processing time and the number of pages.

Applicationa and Database Design Change (1/4)

- Found an online package that runs over 200K times/ Day and consumes the highest CPU time daily.

- Package, whenever executed, had the **same number of OPEN /FETCH/CLOSE** statements. Not matching the number of packages.

- This critical package had an SQL that consumes high CPU and also had the maximum get pages per transaction

- The online transaction, which is a contributing cause for the Application slowdown, was highly impacted because of the package when it processes a large account.

Applicationa and Database Design Change (2/4)

- Create a new table to maintain 1 record per key, tell validity
- Create 2 DB2 triggers to maintain this new table
- Create an external action block/COBOL program to read this new table
- Modify Package to call this new external action block

Applicationa and Database Design Change (3/4)

DATE	No:OF CICS TRANSACTIONS	Avg CPU TIME
10-Feb	214,221	0.031
11-Feb	165,600	0.035
12-Feb	170,354	0.035
13-Feb	169,714	0.069
16-Feb	195,794	0.038
17-Feb	220,464	0.045
18-Feb	167,667	0.034
19-Feb	175,796	0.036
Change Imp- 20-Feb	175,363	0.019
23-Feb	229,510	0.019
24-Feb	205,796	0.020

Average CPU Time before tuning 0.035
Average CPU Time after tuning 0.019

"Average CPU Time for a transaction improved by almost 40%"

Achieved Annum saving of AUD$170,000

Applicationa and Database Design Change (4/4)

In online transactions, it is essential to limit the amount of data retrieved to a manageable level. People do not read hundreds of pages online!

Other Common Application design problems

- ✓**Singleton SELECT in loop**
- ✓**Do not code “program” joins - Allow SQL do the work when possible**
- ✓**Use multi-row fetch – Many rows to be processed**
- ✓**A nested loop join on large tables, without an index on the joining column, is a sin.**

Database Design-Index Columns Swap (1/4)

- A job was consuming an average of 800 CPUsec per day.
- CPU Consumption was higher whenever there was high input volume.
- Found the job is going for an index scan.
- The corresponding base table already had enough indexes; adding more indexes might slow down the performance of data change SQL, as the table had ample changes to it every day.
- Analysed the available index usage.
- Found all other packages using that index, had a match in column 2.

Database Design-Index Columns Swap (2/4)

- ➢ **Found many of the SQL that use this index have a reference to column 2 of the index.**
- ➢ **Suggested to swap the index columns.**
- ➢ **Checked the performance of the SQL before and after SQL change.**

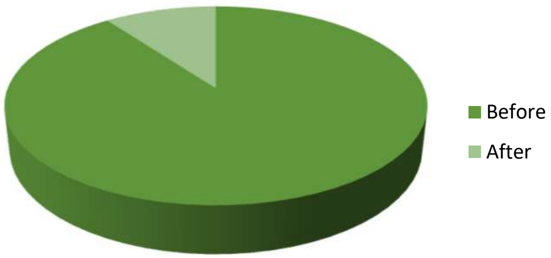
```
Before the index change
                                    Nbr of    --CPU Time--      --Svc Time--
Seqno  Name     Stmt# SQL Function SQL Calls  Total   Mean      Total   Mean

S00047 PACKG031   126 UPDATE            476  120.38   0.25290  198.04   0.41606
S00044 PACKG021   143 SELECT            478    74.22   0.15528  112.98   0.23636
S00043 PACKG011   195 SELECT            477    56.42   0.11829   85.28   0.17878
S00032 PACKG001    95 SELECT            415    50.55   0.12181   69.43   0.16731


After Index Change

                                    Nbr of    --CPU Time--      --Svc Time--
Seqno  Name     Stmt# SQL Function SQL Calls  Total   Mean      Total   Mean
S00045 PACKG031   126 UPDATE              7    0.00   0.00093    0.24   0.03564
S00042 PACKG021   143 SELECT              8    0.00   0.00008    0.00   0.00024
S00041 PACKG011   195 SELECT              8    0.00   0.00019    0.07   0.00931
S00033 PACKG001    95 SELECT             59    0.00   0.00008    0.32   0.00546
```

Existing INDEX columns	Altered INDEX columns
INDXCOL1_ID	INDXCOL2_ID
INDXCOL2_ID	INDXCOL1_ID

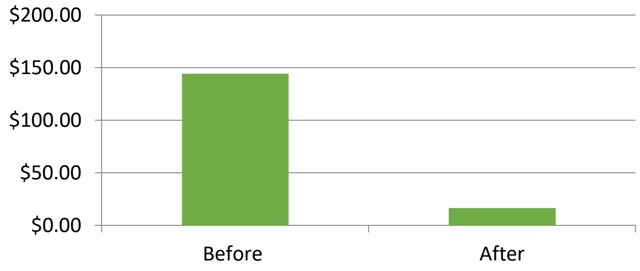
Database Design-Index Columns Swap (3/4)

Date	CPU	
26th Jun	848.70	Before Fix
27th Jun	791.90	Before Fix
28th Jun	559.00	Before Fix
30th Jun	96.30	After Fix
1st Jul	34.50	After Fix
2nd Jul	85.60	After Fix

Daily CPU Usage of the Job



Daily Job Cost



Minimum Per Annum saving will be AUD$58,500

Database Design-Index Columns Swap (4/4)

✓ **Avoid an index scan (Index with match Column zero is bad)**

✓ **Create an index on foreign key columns**

****General Considerations for Database Tuning****

- ❖ Convert to UTS
- ❖ Choose the right partitioning key
- ❖ Fix cluster index
- ❖ Add index-only access
- ❖ Avoid Db2 Sort if possible -Create Matching Indexes
- ❖ Remove redundant indexes
- ❖ Enable compression
- ❖ Clustering and Data Organisation – REORG selective Objects that improve the performance

DB2 Buffer Pool & EDM Pool Tuning (1/6)

- **Started with BP analysis -Found that IO on BP8K0 was around 40% on SYS1, and it is around 25% of overall DB2 IO.**
- **Buffer Pool BP8K0 is mainly used by the system directory table spaces.**
- **The PT(Package Table) Load was around 35% during business hours, which signified that there was something wrong in the system.**
- **PT load should be maintained below 10% for a healthy system.**
- **The size of the EDM Skeleton Pool was the reason for the high IO rate on BP8K0.**
- **The size of the packages was increased in V8, and it was found that the EDM Skeleton pool size did not change in the system.**
- **Found a similar issue in other production DB2 subsystems, too.**

DB2 Buffer Pool & EDM Pool Tuning (2/6)

- Found a similar issue on other production DB2 Sub-systems
- Found that IO on BP8K0 was over 75% on SYS2, and it is around over 10% of total DB2 IO on the application.
- Found that IO on BP8K0 was over 75% and it is around over 45% of total DB2 IO on SYS3 (Supported by Different Vendor)
- Found that IO on BP8K0 was over 75% and it is around over 65% of total DB2 IO on SYS2 (Supported by Different Vendor).

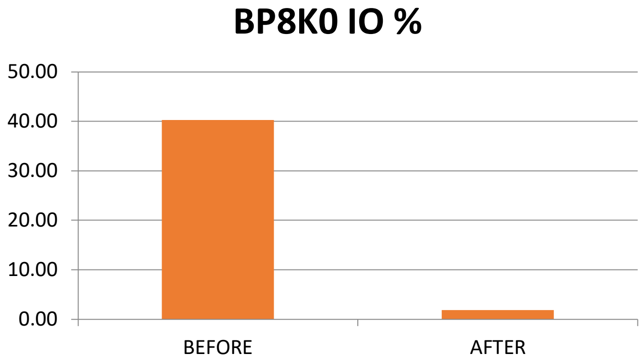
DB2 Buffer Pool & EDM Pool Tuning (3/6)

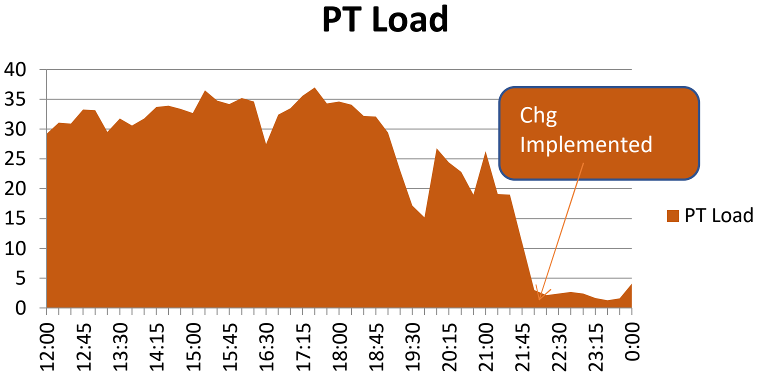
- The size of EDM_SKELTON_POOL was doubled using DB2 ZPARM parameter.
- The size of the system directory buffer pool BP8K0 was also requested to be increased three times.
- IO rate on the System directory Buffer pool was higher than any application buffer pool IO nearly 90% of the time.
- Fixing would result in Average 20% CPU saving in the Db2 Subsystem.

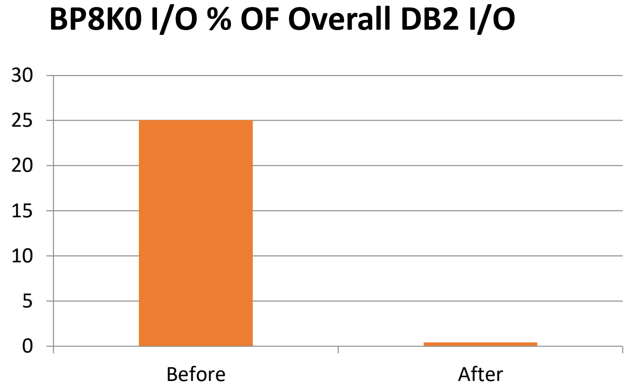
2026 Australia Db2 Tech Conference

DB2 Buffer Pool & EDM Pool Tuning (4/6)

#IDUGDb2





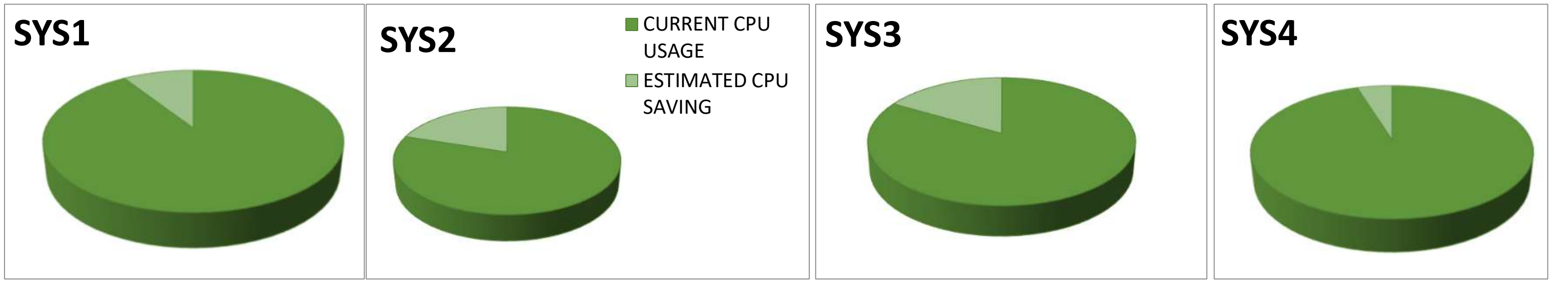


Minimum Per Annum saving will be 3,250,000 CPUSec

Dollar Saving per Annam approx. AUD$ 225,000

DB2 Buffer Pool & EDM Pool Tuning (5/6)

APPLICATION	DB2 IO SAVING %	EST CPU SAVING %	CURRENT CPU USAGE DAILY (CICS+DB2)	ESTIMATED CPU SAVING DAILY	ESTIMATED CPU SAVING PER ANNUM (ONLINE)
SYS1	25	10	108,000	10,800	**3,240,000**
SYS2	65	25	75,600	18,900	**5,670,000**
SYS3	45	20	39,600	7,920	**2,376,000**
SYS4	10	5	55,000	2,750	**825,000**



Projected Minimum Per Annum saving will be over 12,000,000 CPUSec

Projected Minimum Per Annum saving Approx. AUD$ 840,000

DB2 Buffer Pool & EDM Pool Tuning (6/6)

What Happens If EDM Pools Are Undersized?
When DBD / PT / PL loads miss in memory:

➞ Db2 performs **synchronous I/O**
➞ Threads wait for metadata
➞ CPU increases due to reloads
➞ SQL appears “**slow**” even though access paths are fine
➞ Performance issues become **intermittent and hard to diagnose**

Other key things to monitor

Buffer pool hit ratio
&
RID Pool failures

Area	**Key Metrics**
DBD Pool	Hit ratio, DBD loads, reclaims
PT Load	Package load count, reload frequency
Dynamic Statement Cache	Avoiding repetitive PREPARE calls
Overall EDM	Pool usage, Monitor Lock/Latch counts

✓ Keep **frequently executed packages resident/reuse database connections/avoid Prepare**

IDUG

2026

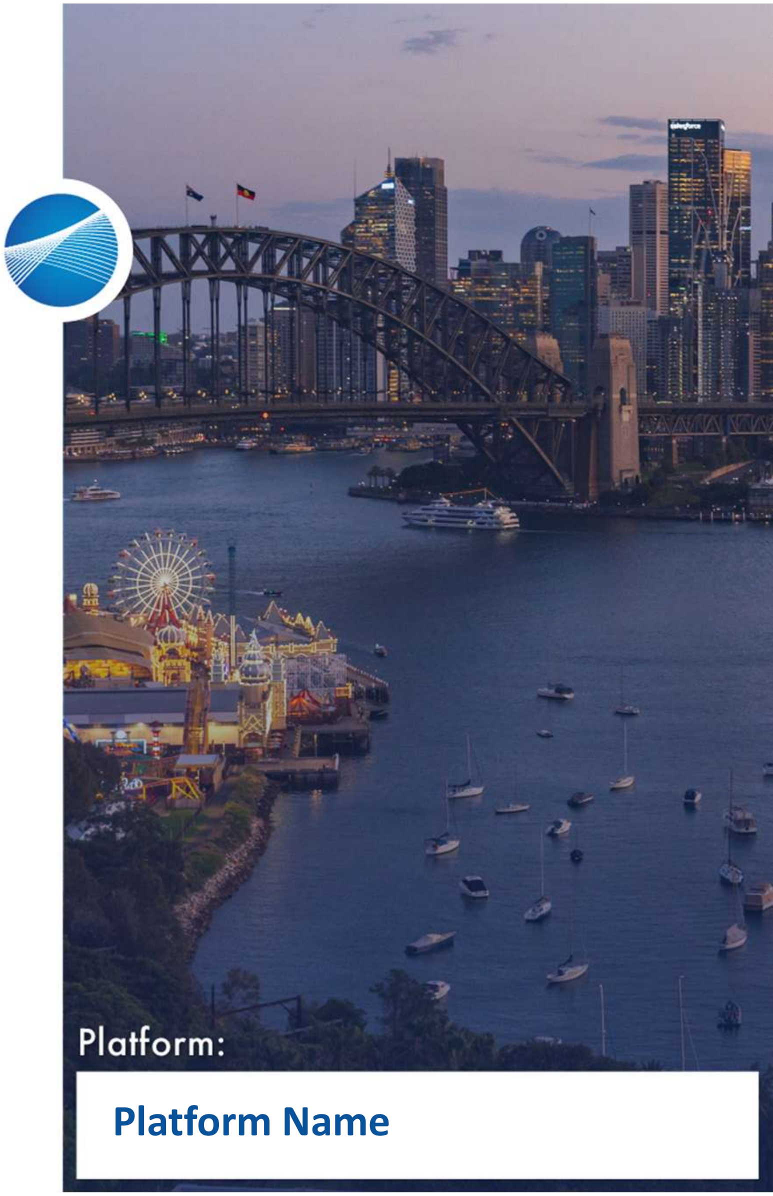
Sydney | March 16 - 18

AU Db2 TECH CONFERENCE

Db2 for z/OS Tuning Techniques: Practical Strategies for Maximum Performance

Devanand Karunakaran, BMC Software
Contact: Devanand_Karunakaran@bmc.com

Session Code: A04



Platform:

Platform Name

IDUG

2026 Australia **Db2** Tech Conference

REBIND-Performance Tuning

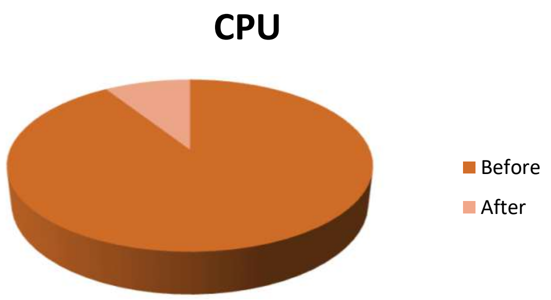
- **Identified that the BMP Job was CPU-intensive and was using a high percentage of DB2 Time .**
- **Identified the SQL that consumed more CPU and issued more get pages.**
- **Found the package had SQL with predicates that matched the index, but the DB2 was using Table space Scan.**
- **Just an explain confirmed DB2 Optimizer was picking the index too.**

REBIND-Performance Tuning

- One of the simplest fix which resulted in huge Savings.
- Reason could be that when the original implementation was done years back, the table did not have any rows or sufficient rows for the DB2 optimizer decided to go for a table space scan

```
                                    Nbr of    --CPU Time--      --Svc Time--
Seqno  Name      Stmt# SQL Function SQL Calls  Total    Mean       Total    Mean

S00119 FCD3DLKA    399 OPEN              230  198.54   0.86324  244.61   1.06355 Before Fix.

                                    Nbr of    --CPU Time--      --Svc Time--
Seqno  Name      Stmt# SQL Function SQL Calls  Total    Mean       Total    Mean

S00166 FCD3DLKA    399 OPEN               23    0.00   0.00024    0.16   0.00706 After FiX
```

REBIND-Performance Tuning



Average 1 week CPU of the job (based on last 4 weeks data)-	11065.87 CPU Sec
With 90% reduction the saving could be	9960.28 CPU Sec
Annul saving (52 Weeks)	**517, 883.95** CPU Sec

Before

JOBNAME-	RUNON	DAY	STAR STOP	ELAPS	CPUSEC
JOB0001	Jul-04	THU	0102_0030	23:28	2847.00
JOB0001	Jul-03	WED	0101_0030	23:29	4133.20
JOB0001	Jul-02	TUE	0042_0030	23:48	1712.20
JOB0001	Jul-01	MON	0055_0030	23:35	2568.40
JOB0001	Jun-30	SUN	0146_0030	22:44	178.10

After

JOBNAME-	RUNON	DAY	STAR STOP	ELAPS	CPUSEC
JOB0001	Jul-25	THU	0040_0030	23:50	255.3
JOB0001	Jul-24	WED	0108_0030	23:22	180.1
JOB0001	Jul-22	MON	0115_0030	23:15	179.2
JOB0001	Jul-21	SUN	0556_0030	18:34	6.4

Minimum Per Annum saving will be AUD$35,000

Database Design-Performance Tuning

- **Found a job taking a lot of CPU to run, and running with 5265.68 CPUSec**
- **Found that it is not a job run through Control-M**
- **Job run twice already and consumed a huge amount of CPU.**

```
ELAPSED-TIME -CPU-TIME-
03:08:02   8653.09S
26:01:58  60002.59S
```

- **This job was taking around 1 CPUsec for every delete**
- **Application team, it is once an activity to purge records from table**
- **To delete 1.2 million records, it might need around 1.2 million CPU sec to complete the delete**

Database Design-Performance Tuning

- The table TBL0002 has delete restrict on TBL0001 , but do not have index on foreign key fields. This makes every single delete on TBL0001 go for a complete table scan on TBL0002, so it takes over 1 CPU Sec to process a single delete statement. It is a bug in the database design; the index should have been created when the foreign key constrain is created between TBL0001 and TBL0002.

- Requested to hold the job until the index is created in production. Creating the new index on TBL0002 and then running the job would save around 1 Million CPU Sec.

- The once-off job to delete 1.2 million records from TBL0001 had consumed around 75K CPU Sec for deleting 65K records of total of 1.2 million

Database Design-Performance Tuning

Before:

```
                                 Nbr of    --CPU Time--       --Svc Time--
Seqno  Name      Stmt# SQL Function SQL Calls  Total    Mean        Total    Mean

S00002 AOOI0001      1 DELETE             302  244.31    0.80898  299.45    0.99156
```

After:

```
                                 Nbr of    --CPU Time--       --Svc Time--
Seqno  Name      Stmt# SQL Function SQL Calls  Total    Mean        Total    Mean

S00002 AOOI0001      1 DELETE         223,628  34.59    0.00015  278.81    0.00124
```

Average 1 CPU for each Delete	1.00 CPU Sec
To delete 1.2 Million records (Estimate)	1.20MillionCPU Sec (In 20 Days)
After change 1.2 Million records Delete	417 CPUSec (in 34 Mins)

Minimum Saving achieved is over 1 Million CPUSec

Projected minimum saving will be AUD$70,000

New Index-Performance Tuning

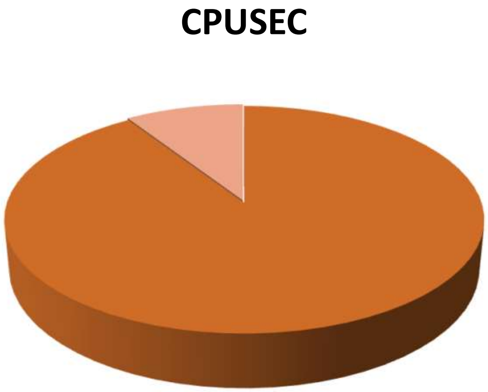
- ➢ **Found a MQ jobs consume high DB2 time.**
- ➢ **Package is going for a Tablespace scan on TBL0001 in production**
- ➢ **Found the number of index on other regions is not matching with production environment**
- ➢ **Production Environment it had just Primary key index alone.**

New Index-Performance Tuning

- We have identified that the performance of the package PACKAGE was not so good, which is used by the MQ Job. This causes the MQ jobs to consume more CPU and some DB2 production contention failures on the TBL0001 table. We need to create a new index on the required columns of table TBL0001 and rebind the package, which will make the query more efficient and reduce the online contention failures that occur on TBL0001.

```
Nbr of    --CPU Time--      --Svc Time--
Seqno  Name     Stmt# SQL Function SQL Calls  Total   Mean      Total   Mean
S00315 PACKAGE   613 SELECT              5    2.97   0.59470    4.77   0.95544 ---Before
S00230 PACKAGE   613 SELECT              4    0.00   0.00023    0.00   0.00122 ---After
```

New Index-Performance Tuning



Minimum Per Annum saving will be AUD$20,000

Online transactions -Performance Tuning

- **The Online transactions are processed through the MQ.**
- **Parallel processing of transactions was also having a few deadlock timeouts, failures, and failed transactions were auto-reprocessed as per the design.**
- **As the daily volume of the transactions was high (above 50K) there was a cost-savings opportunity identified to reduce overheads in the system by providing better access to the data on the database for this specific requirement.**

IDUG
2026 Australia Db2 Tech Conference

Online Transactions -Performance Tuning

#IDUGDb2

- **Changes were made to the Database so that the process directly access active services basically better index access.**
- **Create a new index on Table in addition to existing 4 indexes, as this index will reduce the CPU usage greatly.**

```
                                  Nbr of    --CPU Time--      --Svc Time--
Seqno  Name     Stmt# SQL Function SQL Calls  Total   Mean      Total   Mean

S00063 PACKAGE   764 FETCH             159   38.16   0.24005   54.64   0.34370--Before
S00166 PACKAGE   764 FETCH             141   25.47   0.18067   39.74   0.28184--Before

S00044 PACKAGE   764 FETCH             363    0.03   0.00010    1.98   0.00545-- AFTER
S00178 PACKAGE   764 FETCH             451    0.03   0.00008    1.72   0.00381-- AFTER
```

Online transactions -Performance Tuning

- Reduced the transaction processing time and dead lock time-outs of the Online transaction, which resulted in huge CPU Savings.
- Projected savings of about 140K AUD/year due to this performance tuning. A reduction of 2,086,962 CPU seconds per annum

Minimum Per Annum saving will be AUD$140,000

Performance Issues – After DB2 V10 Upgrade

- Identified the list of jobs below that were performance-impacted after the DB2 version Upgrade.
- All the impacted jobs are dynamic SQL
- Access path change should fix the problem
- Identify the correct access for the SQL and fix it

10 Jobs – Performance degradations

Performance Issues – After DB2 V10 Upgrade

- For 4 parallel jobs, the job was going to the table and processing all accounts instead of going to another table first and processing only the relevant. So changed the DB2 access path with OPTHINT.

- For LOG EXTRACT jobs, the DB2 was not using the index properly, even though all columns of the index were passed from the inner query. So changed DB2 access path with OPTHINT increased to match column of 7 instead of 1.

Performance Issues – After DB2 V10 Upgrade

	CPUSEC (per Week)		CPUSEC Saving
Jobs	Before	After	Per Annam
JOB0001A	14910	207	764556
JOB0001B	14832	209	760396
JOB0001C	14613	211	748904
JOB0001D	15766	216	808600
EXT010X	15458	222	792272
EXT015X	2240	275	102180
EXT0145X	31551	524	1613404
EXT0160X	5879	246	292916
Total			**5883228**

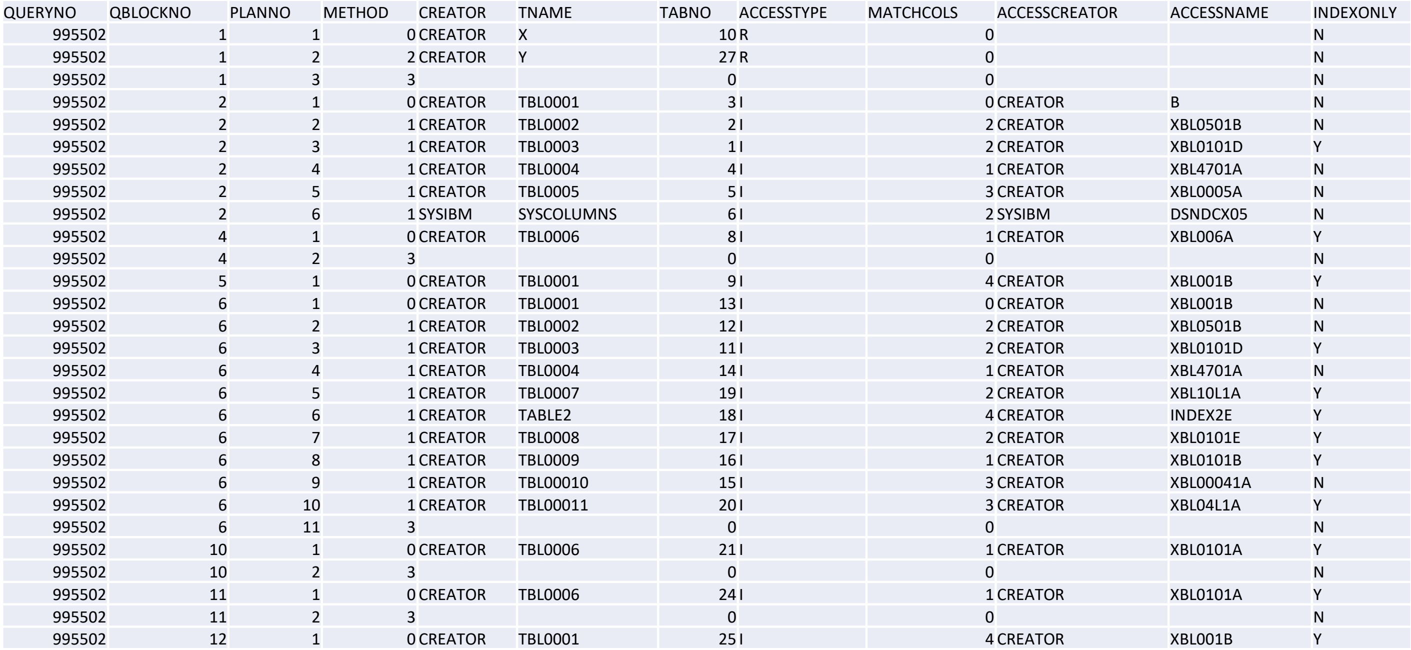
Minimum Per Annum saving will be AUD$400,000

IDUG
2026 Australia Db2 Tech Conference

FIX to Long running monthly job –Performance Tuning

#IDUGDb2

- Application manager requested us to fix the monthly JOB JOB0001, which was running over 36 hours.
- It processes all critical tables of application to generate the monthly recurring valid summary report.
- The job was doing around 400 million get pages to generate the report.
- The job was taking over 11500 CPUSEC and running over 36 hours.



QUERYNO	QBLOCKNO	PLANNO	METHOD	CREATOR	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSCREATOR	ACCESSNAME	INDEXONLY
995502	1	1	0	CREATOR	X	10	R	0			N
995502	1	2	2	CREATOR	Y	27	R	0			N
995502	1	3	3			0		0			N
995502	2	1	0	CREATOR	TBL0001	3	I	0	CREATOR	B	N
995502	2	2	1	CREATOR	TBL0002	2	I	2	CREATOR	XBL0501B	N
995502	2	3	1	CREATOR	TBL0003	1	I	2	CREATOR	XBL0101D	Y
995502	2	4	1	CREATOR	TBL0004	4	I	1	CREATOR	XBL4701A	N
995502	2	5	1	CREATOR	TBL0005	5	I	3	CREATOR	XBL0005A	N
995502	2	6	1	SYSIBM	SYSCOLUMNS	6	I	2	SYSIBM	DSNDCX05	N
995502	4	1	0	CREATOR	TBL0006	8	I	1	CREATOR	XBL006A	Y
995502	4	2	3			0		0			N
995502	5	1	0	CREATOR	TBL0001	9	I	4	CREATOR	XBL001B	Y
995502	6	1	0	CREATOR	TBL0001	13	I	0	CREATOR	XBL001B	N
995502	6	2	1	CREATOR	TBL0002	12	I	2	CREATOR	XBL0501B	N
995502	6	3	1	CREATOR	TBL0003	11	I	2	CREATOR	XBL0101D	Y
995502	6	4	1	CREATOR	TBL0004	14	I	1	CREATOR	XBL4701A	N
995502	6	5	1	CREATOR	TBL0007	19	I	2	CREATOR	XBL10L1A	Y
995502	6	6	1	CREATOR	TABLE2	18	I	4	CREATOR	INDEX2E	Y
995502	6	7	1	CREATOR	TBL0008	17	I	2	CREATOR	XBL0101E	Y
995502	6	8	1	CREATOR	TBL0009	16	I	1	CREATOR	XBL0101B	Y
995502	6	9	1	CREATOR	TBL00010	15	I	3	CREATOR	XBL00041A	N
995502	6	10	1	CREATOR	TBL00011	20	I	3	CREATOR	XBL04L1A	Y
995502	6	11	3			0		0			N
995502	10	1	0	CREATOR	TBL0006	21	I	1	CREATOR	XBL0101A	Y
995502	10	2	3			0		0			N
995502	11	1	0	CREATOR	TBL0006	24	I	1	CREATOR	XBL0101A	Y
995502	11	2	3			0		0			N
995502	12	1	0	CREATOR	TBL0001	25	I	4	CREATOR	XBL001B	Y

FIX to Long running monthly job –Performance Tuning

- Application Team confirmed the need to process all the tables to get the expected report.
- Identified the table on which the maximum get page was happening and analysed the reason. And introduce a new index on that table to improve performance.
- Additionally, identified a couple of more conditions that can be added to the predicate to add more filtering factors.
- Did several rounds of iteration testing that came with the optional fix for the job.

JOBNAME-	ELAPS	CPUSEC	Total Get Pages	SQL Change
JOB00001	9:09	8189.20	199101K	With Index -Original SQL - 2 new condition to improve the match column on TBL0008 and TBL0007
JOB00001	10:29	12808.00	279775K	With new Index
JOB00001	19:51	14075.00	387562K	Baseline

FIX to Long running monthly job –Performance Tuning

- Fix implemented reduced the run time of the job by 75% and the CPU usage by 35%.

- The job started completing within 10 hours.

```
JOBNAME- JOBID- RUNON DAY STAR STOP ELAPS CPUSEC
JOB0001 J51295 OCT15 WED 2316_0844  9:28  7845.00   →  After Fix
JOB0001|J25114 SEP16 TUE 0121_1353 36:32 11735.00
JOB0001|J25197 AUG15 FRI 2311_1122 36:11 11497.00
```

Minimum Per Annum saving will be AUD$ 3,000

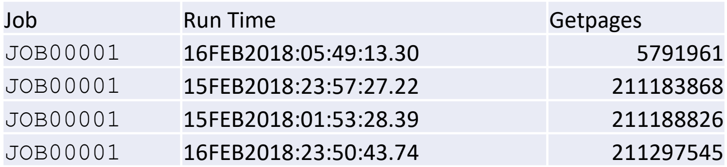
Long Running –High CPU usage batch Job

- The SQL was processing 2 very large tables and a medium Table.
- DB2 Access path showed it started with the medium table and had processed 2.5 million records
- The Job used to run for over 10 hours on a few days when the CPU is unavailable.
- Found the package takes all the time on the Fetch statement.
- The SQL did not have any variables whose results would change. So why did the Fetch take so much time was analysed.

```
JOBNAME- JOBID- RUNON DAY STAR STOP ELAPS CPUSEC
JOB00001|J11907 FEB14 WED 2131_0153  4:22 1098.6
JOB00001|J32473 FEB13 TUE 2132_0202  4:30 1062.9
JOB00001|J22134 FEB12 MON 2130_0003  2:33 1009.0
JOB00001|J11503 JAN26 FRI 2131_1313 15:42 1280.2
JOB00001|J31419 JAN19 FRI 2143_1520 17:37 1261.8
```

Long Running –High CPU usage batch Job

- The number of get pages is very high, as the FETCH is not efficient
- Increase the array size in the program, which reduces the number of fetches it has to do.
- Avoiding additional reprocessing of the SQL statement resulted in good savings.



Job	Run Time	Getpages
JOB00001	16FEB2018:05:49:13.30	5791961
JOB00001	15FEB2018:23:57:27.22	211183868
JOB00001	15FEB2018:01:53:28.39	211188826
JOB00001	16FEB2018:23:50:43.74	211297545

Long Running –High CPU usage batch Job

- Application code changed to increase the array size to hold more records

Minimum Per Annum saving will be 362700 CPUSec /Annum.

Index Change-Performance Tuning

- The MQ Job was consuming high CPU .
- Found one of the SQL going for Index scan with match column zero
- The predicate used had column on the index but not the first column.
- Analysed all the packages that uses that index .
- Found Column2 alone used on the impacted package and all other packages that used the index had both the columns in the predicates.

IDUG
2026 Australia Db2 Tech Conference

Index Change-Performance Tuning

#IDUGDb2

- Suggested to swap the index columns.

Current

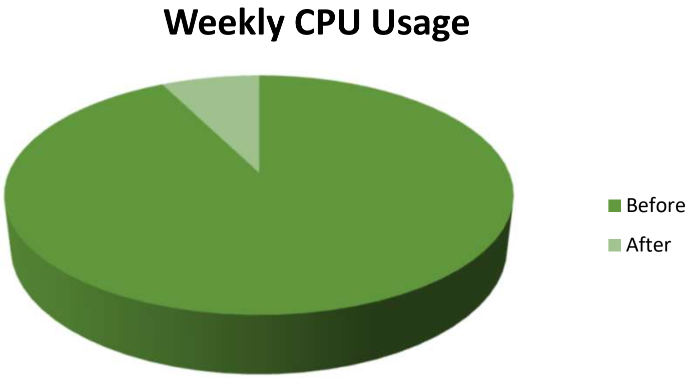
```
CREATE UNIQUE
  INDEX CREATOR.TABLE01
 ON CREATOR.TABLE01
  (
   COL1 ASC,
   COL2 ASC
  )
```

Alter to

```
CREATE UNIQUE
  INDEX CREATOR.TABLE01
 ON CREATOR.TABLE01
  (
  COL2 ASC,
  COL1 ASC
  )
```

Index Change-Performance Tuning

Average 1 week CPU usage of the job (based on last 4 weeks data)-	13607.15 CPU Sec
Average 1 week CPU usage of the job (After Fix)-	1095.80 CPU Sec
Average CPU Saving per week (After Fix)-	12511.35 CPU Sec
Annul saving (52 Weeks)	**650,590.20 CPU Sec**



Minimum Per Annum saving will be AUD$45,000