Welcome to the presentation.
Thank you for taking your time for being here.

Few success stories that are shared in this presentation could be familiar to some of you.
I would still hope that most of you would get something useful out of these lessons that we learnt.
We will have time at the end for questions.
Objectives

- Understand key performance metrics in a DB2 DPF database
- High level overview of configuring KPIs and monitoring Data Warehouse using DBI tools
- Success stories -- Tuning a Data Warehouse using DBI tools and home grown UNIX scripts
- Physical design recommendations in a DB2 DPF environment

Objectives of the presentation that were included when abstract was submitted to IDUG for consideration.
High level Agenda for this presentation.
Agenda could be divided into 2 parts.

In Part 1, we will cover database performance methodology (what to pay attention for) particularly in Data Warehouses and how to monitor using DBI tools.

In Part 2, we will cover challenges we faced and tuning success stories.
Introductions and Background

- Pavan Kristipati
  IS Technical Specialist / Sr. Database Administrator
  Huntington Bank

- Scott Hayes
  President & Founder
  DBI Software

Pavan Kristipati, an IBM certified Database Administrator is currently working as Information Systems Technical Specialist at Huntington Bank in Columbus, Ohio. Pavan could be reached at https://www.linkedin.com/in/pavankristipati or by email at pavan.kristipati@gmail.com

Scott Hayes is President and CEO of DBI Software, a frequent speaker at IDUG conferences, published author, blogger on DB2 LUW performance topics, and the host of The DB2Night Show Edutainment Webinar Series.
Pavan Kristipati, an IBM certified Database Administrator is currently working as Information Systems Technical Specialist at Huntington Bank in Columbus, Ohio. Pavan could be reached at https://www.linkedin.com/in/pavankristipati or by email at pavan.kristipati@gmail.com

Scott Hayes is President and CEO of DBI Software, a frequent speaker at IDUG conferences, published author, blogger on DB2 LUW performance topics, and the host of The DB2Night Show Edutainment Webinar Series.
Brief overview of Huntington Bank and its operations and footprint.

- Founded in 1866 – Close to its 150\textsuperscript{th} year anniversary.
- Six States – Ohio, Michigan, Pennsylvania, Indiana, West Virginia and Kentucky.
- $59 Billion Regional Bank – Headquarters in Columbus, OH.
- Services
  - Commercial, Small Business and Consumer Banking Services
  - Mortgage Banking, Treasury Management and Foreign Exchange Services
  - Wealth and Investment, Trust and Brokerage Services
  - Other Financial Services
- 700+ branches, 1500+ ATMs, Internet and Mobile Banking
- 11000+ employees
Data Warehouse Database environment at Huntington Bank on which the monitoring/tuning efforts shared in this presentation.

- IBM Smart Analytics System 5600
- DB2 Product name: InfoSphere Warehouse Enterprise Edition
- DB2 9.7 fp7
- SUSE LINUX 10.4
- DPF – Database Partitioning Feature
- 1 Admin node and 3 Data nodes (hosts)
- 4 Logical partitions on each Data node
- Total -- 1 (admin) + 12 data (3*4) = 13 partitions
- 1 Standby node – HA solution (TSAMP)
Enterprise Data Warehouse at Huntington

- Essential hub for high quality Data and Information for the Bank
- Provides one view of the customer – this was a big deal!
- 50+ source systems writing into Data Warehouse
- SAS Debit Card fraud analytics, Consumer, Auto loan, Mortgage data, Federal Govt. Reporting etc.
- ~1200 tables and growing
- Database size ~ 5 TB and growing (~300 GB a month)
- ~1200+ ETL jobs per day

Details about Enterprise Data Warehouse (EDW) at Huntington Bank.
As EDW provides 360 degree view of the customer, its database performance and availability is of paramount importance to the Bank.
Huntington Bank purchased an enterprise license for DBI tools to manage performance of DB2 LUW databases. These tools are widely used in performance management of both DW and OLTP databases.

At Huntington, DBAs use a combination of DBI tools and home grown UNIX scripts to monitor and manage DB2 LUW databases.
Scott Hayes

- About DBI
- Data Warehouse Performance Best Practices

DBI Software is Your trusted partner for Breakthrough DB2 Performance Solutions that DELIVER INVALUABLE RESULTS for Organizations having the most Demanding Requirements and Discriminating Preferences.

Scott Hayes began working with DB2 on distributed platforms at V1 DB2/6000. 22 years later, after having worked with 100’s of customers and clients around the world, he has learned a couple of things about what makes Data Warehouse Databases run fast.
While transactional databases focus on indexes and optimized synchronous I/O, for data warehouses you need to optimize parallel I/O and prefetching.

While many people pay attention to having data evenly distributed across the partitions, it is also important, maybe more important, to make sure that PROCESSING is evenly distributed across your partitions. If Parts 1, 2, 3, and 4 are working really hard, and 9, 10, 11, and 12 are sitting on their hands, then you’ve got an opportunity for improvement. Logical Read COST (LREADs/Transactions) is a good measurement to evaluate.
Data Warehouse Performance Best Practices 2

- Beware of Overflows
  - Overflows = Double the I/O !!!
  - Catalog Tables! SYSCAT.INDEXES, TABLES, COLUMNS...
- Monitor & Trend Read & Write I/OLatency Times
  - Prefetch/Async Read I/O is normally VERY fast, <3ms (Sync I/O <10ms)
    - Async Read % (ARP) normally 60-90%, Sync Read % (SRP) 10-20%
      - For Tablespaces with ARP>60%, assign to ASYNC BP's w/ NUMBLOCK_PAGES
    - High Async Write % desirable
      - TEMPSPACE is often a bottleneck

DW DBs do plenty of I/O without the unnecessary burden of doing double the I/O due to overflows. If your ETL processes do UPDATES that increase lengths of VARCHAR columns, you can end up with a lot of overflow rows – need to REORG, or cleanup the Overflows in V10.5. DB2 catalog tables are frequent victims of high Overflows. REORG & RUNSTATS – love your catalog tables!

db2set DB2_PARALLEL_IO=*

Randomly read tablespaces can be assigned to a Random bufferpool, including small hot lookup tables.
Most tablespaces will have scans – put into an ASYNC bufferpool and don’t forget NUMBLOCK_PAGES set to ~3% of Pool size
Place your TEMPSPACE on storage paths separate from tablespaces & indexspaces, if you can

We have seen some customers be able to identify storage system problems by detecting abnormally high Read/Write latency times in DB2.
You still need indexes in a DW! Good ones! It’s mind boggling, but we’ve seen some DW databases run queries twice as fast, or faster, just by putting a needed index on a 10MB table! Yes, small tables will likely remain resident in a bufferpool, but you will suck the life out of your CPUs by scanning zillions of rows in memory!

Low cardinality indexes can be very adverse to LOAD & ETL performance because of high cost of updating long rid list chains.
Up next

Pavan Kristipati
High level agenda for the remainder of the presentation. We will have few minutes for questions at the end.

- Data Warehouse KPI monitoring using DBI tools
- Tuning Success Stories
- Physical Design Recommendations
Data Warehouse KPI monitoring using DBI tools
Out of the box, DBI's Brother Hawk has 100+ pre-defined alerts. This helps DBAs to start to keep tabs on database performance and get alerted when thresholds are crossed as soon as the toolset is installed.

Squelch feature gives option to snooze an alert until a custom defined interval (for each alert type). Repeated alerts for the same alert type will indicate on-going problem.

**ORMS** = Overall Disk Read Time in milli seconds

**OWMS** = Overall Disk Write Time in milli seconds
Snapshot of some of the KPIs that Huntington DBAs use to manage EDW.
Lock waits, Log Used % would be more useful in OLTP environments.
Sort overflow, Statement CPU %, IREF, % Table overflows are few KPIs that we keep tabs on to monitor for trends.
Example of how to configure an alert in DBI’s Brother Hawk.
Double click on alert of interest and schedule (start, end times and days of the week) a window to monitor.
Thresholds could be picked for alarm / Warning / Information. SMTP alerts could be sent and / or OS / Database commands could be run when a threshold is breached. In one of the slides, there is an example of how we take an action when table overflows are more than 3%.
Brother Hawk’s Alerts Console to identify High/Medium/Low priority alerts and identify any action that is needed to be taken.
We are now at 2\textsuperscript{nd} part of the presentation in which I am going to share 4 tuning success stories on Huntington’s DPF database.
We started noticing (getting alerted for) CPU spikes during ETL cycle duration when new code was migrated to production. Besides new code in production, there were power users who contributed to CPU spikes in production. This ad hoc activity was unpredictable unlike ETL activity. CPU utilization was frequently ~70% and sometimes reaching 90+%. DBI’s Panther lets you focus on a particular time interval for more focused analysis.
DBI’s Brother Panther allows to zoom on an area of interest to get focused look at CPU utilization. There is also an option to view the ‘statement activity’ during the timeframe selected.

Instead of doing this, what we did was we wanted to take a holistic look and find out what are our problematic tables over 24 hour timeframe.
This slide shows % Rows Read and Absolute value of Rows Read at table level for top 10+ tables. Data is sorted by % Rows Read.

As you notice some values for Rows Read were in Billions of rows and that was over 24 hour period. High # for rows read indicates presence of table scans which are usually costly especially when tablescans are done multiple times.

Our original goal was to free up CPU cycles to avoid potential costly hardware upgrades. Top contributors to CPU cycles are tables with too many tablescans.

As noticed in this slide, top 5 tables use 76% of the total rows read in a 24 hour time period.
Relation between Rows Read and CPU Cycles

- Rows Read is the number of rows that DB2 picked up from the data pages and evaluated for potential inclusion in the result set → Tablescans !! → Burn CPU cycles
- Rows Read is not incremented when Index Only Access is used.

Avg. IX Logical Reads = 0 indicates absence of index scans (and hence presence of table scans). High values for rows read lead to high CPU consumption mostly due to costly table scans.
Once we identified tables that contributed to high CPU usage, next task was to identify statements that were actually running against them.

One of the most important KPIs for a statement is Index Read Efficiency (IREF) for a statement defined as the number of rows read by DB2 for potential inclusion for each one row that was selected.

\[
IREF = \frac{\text{No. of rows read}}{\text{No. of rows selected}}
\]

High IREF indicates table scans due to missing indexes (and hence the term) or possibility of leaf page scans due to bad indexes indicating a need for better quality indexes.

After running explain plans (DBI tools allow to do this), we noticed there were few missing indexes for these statements that were contributing to 76% of CPU usage.
For each of the costly statements, right click on the statement and click on “Generate Explain” to take a look at DB2’s Explain Plan.

We noticed Table Scans (as guessed in previous slides) for most of the statements.

Clicking on “Design Analysis” on the top left of the screen gave access to DB2’s Advisor and its analysis as shown in next slide.
This slide shows the output of DB2 Advisor and its analysis.
As evident from the output, for one of the statements, we were able to reduce the cost of the query by ~64% by creating 4 indexes.

There is an option to either ‘save’ the output or ‘Run’ (create indexes right then and there). In production environment, we saved the report and scheduled index creation activity after obtaining approvals.
Just to recollect…This slide shows the CPU usage before tuning activity was performed.
After 4 simple steps, we noticed the Avg. CPU consumption is ~50% a big drop from earlier ~75-80% !!
Steps taken to reduce CPU Utilization

Step 1: Identify top tables driving CPU utilization
- High % Rows Read, High Rows Read / Tx

Step 2: Identify Statements driving workload to tables from step 1
- High IREF (Index Read Efficiency)
- High% CPU Time

Step 3: Look at Explain Plans and Design Advisor recommendations

Step 4: Create High Quality (Good) Indexes

Result: Reduced CPU Utilization and faster Query Run time

4 steps taken to reduce CPU utilization. While it might appear to be like finding few needles in haystack, with step 1, we tend to reduce the haystack size to manageable level. Once Tables with high % of rows read are identified, next step is to identify statements and analyzing cost, creating good quality indexes.
This is challenge #2 in which we had to tune a SAS workload.
This slide shows Brother Panther’s Work Load analysis by user over 24 hours. SASRRPD (2nd from top) is the USER that runs SAS work load.

As noticed, after tuning the workload, the % CPU time taken by SASRRPD user is < 7% with Index Read Efficiency < 1!

This is a result of tuning effort that could be easily shared with management!

While we could potentially capture workload using native db2 tools like event monitors at the command line, it is very difficult to gauge the relative cost of the workload at user ID or application ID level.
Before incremental backups were implemented, backup policy was to take full online backups nightly.

Before switching to combination of full + incremental backups, we wanted to identify opportunities to tune full online backup. Original backup runtimes were 10-12 hours and were unacceptable.
Root Cause for Long Backup runtimes

- Observation – Backup was slower on few partitions compared to others
- Root Cause(s) in database layer
  - Partitions were heavily skewed in size
    - Faster growth of some of the tables than anticipated -> data skew
    - Partition 0 size -- ~150% bigger than other partitions
    - Data skew at table level -- Hash keys were picked to collocate tables
  - Few tablespaces were very large (~1000x) compared to others

Backup progress for each partition could be monitored using DB2’s “list utilities show detail” command.
How we fixed Longer Backup runtime by making Physical Design Changes - 1

- Solution – Physical Design change to fix skew

- **Step 1:** Hash large tables on Partition 0 (~150% bigger)
  - Why? Backup is as slow as the largest partition
  - SQL to identify tables with card > 1 M on Partition 0

```sql
select char(tabschema, 15) as schema, char(tabname, 70) as tabname, card, char(tab.tbspace, 20), DBPGNAME from syscat.tables tab, syscat.tablespaces tsbsp where tab.tbsp=tsbsp.tbsp and card > 1000000 and dbpgname = 'SDPG' with ur -- (SDPG = Single Data Partition Group aka Partition 0)
```

What we did?
- Identified 61 candidate tables for hashing
- Created tables with hash keys (Primary Key Columns)
- Used Load from cursor
- Renamed tables

An alternative to using load from cursor is to use ADMIN_MOVE_TABLE stored procedure
Step 2: Move large tables into their own tablespaces
  - Why? Very large tablespaces (compared to others) are bottleneck for backup
  - SQL to identify candidate tablespaces in notes
    - Prints tablespace, size, table_count in desc

What we did?
  - Created new tablespaces and tables
  - Used load from cursor or ADMIN_MOVE_TABLE to move tables into new tablespaces

with
temp1 as
  (select TBSP_NAME as tbsp_name, sum(TBSP_USED_SIZE_KB/1024) as tbsp_size
   from sysibmadm.tbsp_utilization
   -- WHERE DBPGNAME in ('PDPG', 'ALLPG')
   group by TBSP_NAME)

temp2 as
  (select t1.tbspace as tbsp_name, count(*) as tbl_cnt
   from sycscat.tables t1, syscat.tablesspaces t2
   where t1.type='T' and t1.tbspace=t2.tbspace
   -- and t2.ngname in ('ALLPG', 'PDPG')
   and t1.tbspace <> 'DWEDEFAULTCONTROL'
   group by t1.tbspace
   having count(*) > 1)

select char(temp1.TBSP_NAME, 20), temp1.tbsp_size, temp2.tbl_cnt
from temp1, temp2 where temp1.tbsp_name=temp2.tbsp_name
order by temp1.tbsp_size desc fetch first 100 rows only with ur
How we fixed Longer Backup runtime by making Physical Design Changes - 3

- **Step 3**: Reduce High Water Mark (HWM) wherever possible
  - Why? Backup runs until High Water Mark (HWM)
  - Identify top candidates to lower HWM
    - SQL 1 in notes – Identifies tablespaces with > 1 GB HWM reduction opportunity
    - SQL2 in notes (if curious) – How much HWM (MB) would be reduced in each container (DB2 automatically calculates and reduces)
  - How to reduce HWM?
    - ALTER TABLESPACE $tbsp LOWER HIGH WATER MARK
  - **200 GB reduction in HWM**

**SQL1 -- Identify tablespaces with > 1 GB HWM reduction opportunity:**
```sql
SELECT CHAR(TBSP_NAME,18) AS TBSP_NAME,
       SUM((TBSP_PAGE_TOP - TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024/1024) as TO_BE_REDUCED_SPACE_GB
FROM SYSIBMADM.TBSP_UTILIZATION
GROUP BY TBSP_NAME
HAVING SUM((TBSP_PAGE_TOP - TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024/1024) > 1
ORDER BY 2 DESC
FETCH FIRST 100 ROWS ONLY WITH UR;
```

**SQL 2 – Identify how much space (MB) would be reduced from each container:**
(if you are curious to find out how much HWM in MB would be reduced by DB2)
```sql
SELECT
       char(TBSP_UTIL.TBSP_NAME,20) AS TABLESPACE,
       char(CONTAINER_NAME,50) as CONTAINER,
       TBSP_UTIL.DBPARTITIONNUM as PARTITION,
       TBSP_PAGE_TOP                       as HWM,
       (TBSP_FREE_SIZE_KB/1024)            as FREE_MB,
       (TBSP_PAGE_TOP - TBSP_USED_PAGES)*TBSP_PAGE_SIZE/1024/1024 as HWM_REDUCTION_OPPORTUNITY_MB
from SYSIBMADM.TBSP_UTILIZATION TBSP_UTIL,
     SYSIBMADM.CONTAINER_UTILIZATION CONT_UTIL
WHERE TBSP_UTIL.TBSP_NAME=CONT_UTIL.TBSP_NAME AND
     TBSP_UTIL.DBPARTITIONNUM=CONT_UTIL.DBPARTITIONNUM
AND
     TBSP_UTIL.TBSP_NAME='$TBSP' order by TBSP_UTIL.DBPARTITIONNUM with ur;
```
How we fixed Longer Backup runtime by making Physical Design Changes - 4

- **Step 4:** Fix skew at table level
  - Custom script (in crontab) records data skew info for all tables
  - Stored procedure "estimate_existing_data_skew"
  - Stored procedure "estimate_new_data_skew"
  - New Hash keys (PK) for 2 large tables
  - In notes
    - Info Center URL to download Stored Procedures
    - Usage with an example


Example from IBM Info Center:
$db2 "set serveroutput on"
$db2 "CALL estimate_existing_data_skew('TPCD', 'SUPPLIER', 25)"
CALL estimate_existing_data_skew('TPCD', 'SUPPLIER', 25)
Return Status = 0
DATA SKEW ESTIMATION REPORT FOR TABLE: TPCD.SUPPLIER
Accuracy is based on 25% sample of data
------------------------------------------------------------------------
TPCD.SUPPLIER
Estimated total number of records in the table: : 19,994,960
Estimated average number of records per partition : 2,499,368
Row count at partition 1 : 1,599,376 (Skew: -36.00%)
Row count at partition 2 : 2,402,472 (Skew: 3.87%)
Row count at partition 3 : 4,001,716 (Skew: 60.10%)
Row count at partition 4 : 2,394,468 (Skew: -4.19%)
Row count at partition 5 : 1,600,028 (Skew: -35.98%)
Row count at partition 6 : 1,599,296 (Skew: -36.01%)
Row count at partition 7 : 2,397,116 (Skew: -4.09%)
Row count at partition 8 : 4,000,488 (Skew: 60.05%)
Number of partitions: 8 (1, 2, 3, 4, 5, 6, 7, 8)
------------------------------------------------------------------------
Total execution time: 20 seconds
Earlier backup duration was in the range of 9 to 13 hours.

After physical design changes, backup runtime is averaging between 5 to 6 hours.

To save disk space on TSM and to reduce impact of backups on workload, combination of Full+Incremental backups were implemented. Because of the physical design changes that were made to help full backups, Incremental backups mostly run in less than 1 hour and average between 30 to 90 minutes on any given day depending on the amount of data that needs to be backed up.
A major question that is left over is if the physical design changes that were done impacted SQL workload in a negative way? DBI’s Panther answers that question.
One exciting feature in DBI’s Panther is that it could do workload comparison between 2 time intervals. In this slide, the workload comparison is done between before and after Physical design changes. 24 hour time interval was taken as comparison and as you notice good physical design changes mostly help SQL queries to run faster and DBI’s Panther provides the proof for this!!
On a given day, about 20% of our Data Warehouse was being “changed” (updates mostly). While REORG operations could avoid costly double IO, fundamental question that remains is which tables to REORG?

Challenge 4: Uncertainty around REORGCHK / REORG

- Quick Background
  - Data Warehouse -- Updates on ~20% of data
    - Example: COLUMNN – VARCHAR(100)
      -- Value “PAVAN KRISTIPATI” updated to “PAVAN KUMAR KRISTIPATI”
      -- Row (wider) doesn’t fit into the same page
      -- DB2 relocates row to new page;
      -- Pointer in the original location to the new location
  - Result – Double IO !!
    - How to fix this?
      - DB2 attempts to read the row from its original location
      - Finds a pointer instead
      - DB2 now reads the row from its new location
Uncertainty around REORGCHK / REORG

- **REORGCHK**
  - Utility to indicate when / what to REORG
  - Formulas based on Table Cardinality, Overflows etc.

- **Limitations with REORGCHK’s analysis**

- **Observation 1:**
  - Analysis based on data sampling of the entire table
  - Criteria used does not work well with large tables
  - Example: In Large tables
    - Older data – Well Organized (No updates after last Reorg)
    - Newer data – Fragmented (due to Updates)
    - Interest in newer data – Accessed more frequently (Could be a small %)
  - Could miss marking large tables to be reorg’d.

REORGCHK is a native IBM DB2’s utility to help to identify which tables to REORG.
Uncertainty around REORGCHK / REORG

- **Observation 2:**
  - Uses one of the two options – Current stats or Update stats
  - Current stats: 24 hours old (AUTO MAINT off) – stats probably no longer valid
  - Update stats: New stats alter SQL access plans for Dynamic SQL

- **Observation 3:**
  - Formula F1
    - based on how big (card) the table is and not how active the table is
    - # Overflows do not get reset after reorg (needs database deactivation !)
  - F1: 100 * overflows / card < 5 (If F1 > 5, mark for reorg)

DBI’s tools give a huge advantage in knowing which tables need to be reorg’d in that the tools help in identifying those tables that were read/accessed in the timeframe of interest.

Taking the traditional approach (reorgchk) does not give this option. Formulae are based on the row size of the table and not how “active” (rows read) the table really is.

SQL to find overflows for tables. Please note that ROWS_READ, ROWS_WRITTEN, OVERFLOW_ACCESSSES values are since database activation time.

```
db2 "select char(tabschema, 20), char(tabname, 40),
sum(overflow_accesses) as total_overflow_accesses, sum(rows_read) as total_rows_read, sum(rows_written) as total_rows_written from sysibmadm.snaptab group
```
by tabschema, tabname order by
total_overflow_accesses desc, total_rows_read
desc fetch first 20 rows only with ur"
**Uncertainty around REORGCHK / REORG**

**Alternative: DBI’s Approach:**
- Based on # **overflows and rows read** in a **specific timeframe**
- % Overflow = \(100 \times \text{overflows} / (\text{Rows Read} + 1)\)

P.S.: SQL in the notes to find out overflows from “database activation time”

DBI’s tools give a huge advantage in knowing which tables need to be reorg’d in that the tools help in identifying those tables that were read/accessed in the timeframe of interest.

Taking the traditional approach (reorgchk) does not give this option. Formulae are based on the row size of the table and not how “active” (rows read) the table really is.

SQL to find overflows for tables. Please note that ROWS_READ, ROWS_WRITTEN, OVERFLOW_ACCESS values are since database activation time.

```
db2 "select char(tabschema, 20), char(tabname, 40), sum(overflow_accesses) as total_overflow_accesses, sum(rows_read) as total_rows_read, sum(rows_written) as total_rows_written from sysibmadm.snapstab group
```
by tabschema, tabname order by total_overflow_accesses desc, total_rows_read desc fetch first 20 rows only with ur"
Candidate tables for REORG

- Tables with % overflows > 3%
- An effective way to avoid double IO
- Pay attention to overflows in “Catalog” tables

Often system catalog tables show up in the list of top tables with overflows.
Using DBI’s Brother Hawk, we reorg tables based on % table overflow values. If % table overflow > 3, table is reorg’d.
DB2 DPF – Top Recommendations at Database level

- Partition groups – SDPG (0), PDPG (1-12) and ALLPG (0-12)

<table>
<thead>
<tr>
<th>Table Card &lt; 1 M</th>
<th>SDPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Card &gt; 1 M</td>
<td>PDPG</td>
</tr>
<tr>
<td>MQTs</td>
<td>ALLPG</td>
</tr>
</tbody>
</table>

- Lookup tables -- heavily used code values are integers (rather than character) -- Faster

- Goal -- Equal workload on all data partitions
  - High Cardinality columns (Primary Key) as hash key
  - Collocate if absolutely necessary for SQL performance

- Goal – Faster backup and recovery times
  - Avoid very big tablespaces – large table in its own tablespace
  - Avoid db partition level skew (best practice for all over-all database performance)
DB2 DPF – Top Recommendations at Database level

- Database Performance
  - Create Good (High Cardinality) Indexes
  - Avoid Bad (Low Cardinality) Indexes – Drop them if they exist. (Ember Crooks in DB2Nightshow on April 18th 2014)
  - Take advantage of MQTs and MDCs
  - Drop duplicate indexes – Index on col1 and Index on col1, col2
  - Optimize Async / Sync IO (For DW, aim for SRP > 25%)
- Use Range Partitioned tables for large tables. Helps in implementing data retention/archival.
- Use Compression on large tables
- Review DB2 Advisor’s output for quality of indexes
Another talk at IDUG NA 2014
C08 - DB2 DPF Tips and Tricks: UNIX, SQL scripts and more for Lazy DBAs

Pavan Kristipati
*Huntington Bank*

Rao Balaga
*Huntington Bank*

Session Code: C08
Wed, May 14, 2014 (02:15 PM - 03:15 PM) | Platform: DB2 for LUW
Thank you for attending. Any Questions?
Pavan Kristipati
Huntington Bank
pavan.kristipati@gmail.com
www.db2talk.com

Scott Hayes
Database Brothers Inc.
shayes@dbisoftware.com

C10 - DB2 DPF Successes: Monitoring and tuning a hybrid IBM InfoSphere Warehouse

Please fill out your session evaluation before leaving!