This presentation will guide DBAs through the process of implementing a historical Performance Database, how to populate with relevant data and how analysis of historical data can prevent performance problems before they occur

Objectives

- What is a PDB (Performance Database)?
  - Description of data collected and tables
- Pre-requisites for a PDB
  - Data Extract, Traces, Storage
- How to Create & Populate a PDB
- Tips to Operate & Maintain a PDB
- Real world scenarios
  - Historical trend analysis and proactive problem solving

What is a Performance Database? Description of data collected and tables

Pre-requisites for populating a Performance Database, Data Extract, Traces, Storage

How to Create & Populate a Performance Database, load jobs, etc

Tips to Operate & Maintain a PDB, reduce volume of data collected, via aggregation

Real world scenarios of historical trend analysis and proactive problem solving

Agenda

- How can History Help?
  - Introducing the PDB
  - What is it...and why do I need one?
- SMF Data Repository
  - Extract
  - Create
  - Populate
- Design (Decisions, Decisions)
  - How Much History
  - Ad-hoc Tables
  - Data Warehouse Location
- Reporting
  - Trend Analysis (Predicting the Future)
  - Proactive Problem Determination

How can history help?

- How can History Help?
  - Introducing the PDB
  - What is it…and why do I need one?
- SMF data Repository
  - Extract
  - Create
  - Populate
- Design (Decisions, Decisions)
  - How Much History
  - Ad-hoc Tables
  - Data Warehouse Location
- Reporting
  - Trend Analysis (Predicting the Future)
  - Proactive Problem Determination

How can history help?

- What is a PDB (Performance Database)?
- Definition: -
  - *"A data store of performance related data, counters and statistics, providing an historical view of application performance, allowing trend analysis and proactive identification of impending performance degradation"*
- Alternative Definition: -
  - *"A database about performance"*

How can history help?

- Reactive Monitoring
  - Real Time Monitors
  - Threshold Alerts
  - Near Time History
- What about Pro-active?
  - Prevent problems before they occur

How can history help?

- What is Normal?
  - Trend Analysis
  - Capacity Planning
  - Central Repository

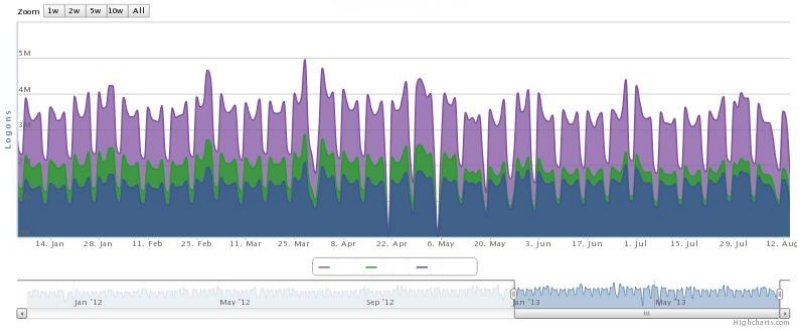
Maybe a
Performance Database?

How can history help?

- Sources of Data
  - SMF
    - Accounting
    - Statistics
  - RMF
    - Transaction Performance
  - Application Tables
    - Transaction Audit
    - MIS

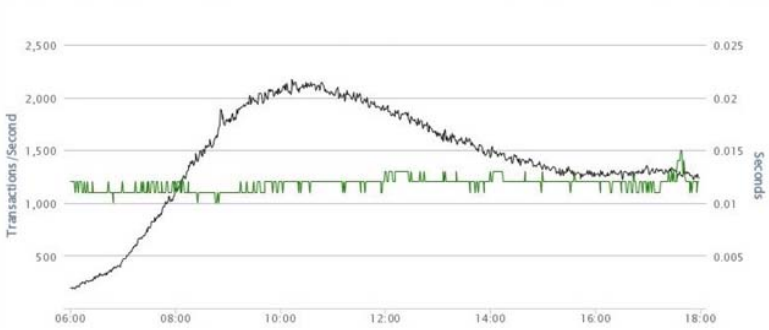
How can history help?

- What data to capture & store?
  - Bespoke/application specific
    - Purchases
    - Customer logons



How can history help?

- What data to capture & store?
  - Transaction Performance from RMF
    - Average elapsed time
    - Transaction rate



How can history help?

- What data to capture & store?
  - Detailed DB2 Performance from SMF
    - DB2 System
    - Buffer Pools
    - Detailed Application
  - Event Driven & Interval Based

Sounds great but…

…I haven't got time to design & create tables
…write SMF extract jobs
…create Load jobs

SMF Data Repository

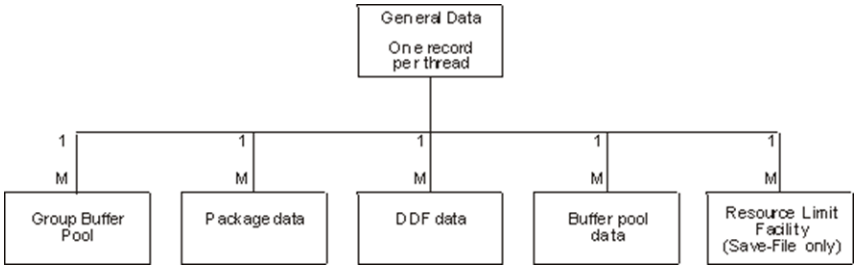
- How can History Help?
  - Introducing the PDB
    - What is it…and why do I need one?
- SMF data Repository
  - Extract
  - Create
  - Populate
- Design (Decisions, Decisions)
  - How Much History
  - Ad-hoc Tables
  - Data Warehouse Location
- Reporting
  - Trend Analysis (Predicting the Future)
  - Proactive Problem Determination

SMF Data Repository

- Statistics Trace data
  - SMF 100 records
  - Statistics Classes 1, 3, 4, 5, 6
  - Interval based
    - Check ZPARMs: - STATTIME, SYNCVAL, SMFACCT, SMFSTAT
- Accounting Trace data
  - SMF 101 records
  - Accounting Classes 1, 2, 3
  - More detail Classes 7, 8, 10
  - Event driven

SMF Data Repository

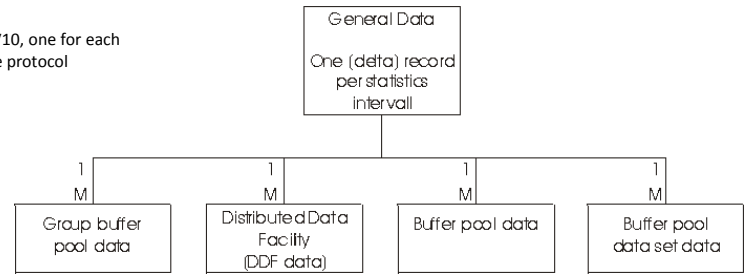
- Accounting Data
  - **General Accounting** - One row per thread
  - **Packages** - One row per package and DBRM executed
  - **BPs/GBPs** - One row per buffer pool used
  - **DDF** - One row per DDF remote location



SMF Data Repository

- Statistics Data
  - **General Statistics Counters** - One row for each Interval
  - **BPs/GBPs** - One row per buffer pool active at interval start
  - **DDF** - One for all remote locations that used DRDA*
  - **Dataset I/O stats** - One row per data set with I/O rate > one per second during interval

*For those not yet on DB2 z/OS V10, one for each remote location still using Private protocol



SMF Data Repository

- Extracting SMF Data
- Bespoke, BMC Mainview, Omegamon DB2PE, Others
- Example will use DB2PE REPORT command
- Two DB2PE choices: -
  - FILE Command results in non-summarised DB2 Load compatible data
  - SAVE Command results in summarised data. Not Load compatible.
- Mainview provides option to summarise data also

SMF Data Repository

- Extracting Data
  - FILE Command results in non-summarised DB2 Load compatible data
  - Example DB2PE FILE command to extract GENERAL, PACKAGE & DDF data to 3 separate datasets

```
GLOBAL
   INCLUDE (SUBSYSTEMID(ssss))
   FROM (yy/mm/dd:hh:mm:ss)
   TO(yy/mm/dd:hh:mm:ss)
ACCOUNTING
   FILE
      DATATYPE(GENERAL)
      DDNAME (ACFILDD1)
   FILE
      DATATYPE(PACKAGE)
      DDNAME (ACFILDD2)
   FILE
      DATATYPE(DDF)
      DDNAME (ACFILDD3)
EXEC
```

SMF Data Repository

- Extracting Data
  - SAVE Command results in summarised data in VSAM file
  - Has to be converted to non-summarised DB2 Load format
  - Utility DGOPMICO will convert VSAM to Load compatible format
  - Example DB2PE SAVE command: -

```
GLOBAL
   INCLUDE (SUBSYSTEMID(ssss))
   FROM (yy/mm/dd:hh:mm:ss)
   TO(yy/mm/dd:hh:mm:ss)
ACCOUNTING
   REDUCE
      INTERVAL(15) BOUNDARY(60)
   REPORT
SAVE
   DDNAME (ACSAVDD1)
EXEC
```

SMF Data Repository

- Creating Tables
  - Create tables in non-production environment
  - Ensure plenty of DASD
  - Non-Application Buffer Pools
  - Think about Indexing strategy
  - How much data to keep?
  - Concurrent streams to minimise Load time
  - Consider Partitioning

SMF Data Repository

- Database Creation
  - RKO2SAMP Dataset

Data	Create Table DDL	Table Name
General	DGOACFGE	DB2PMFACCT_GENERAL
Package	DGOACFPK	DB2PMFACCT_PROGRAM
Buffer Pool	DGOACFBU	DB2PMFACCT_BUFFER
Group Buffer Pool	DGOACFGP	DB2PMFACCT_GBUFFER
DDF	DGOACFDF	DB2PMFACCT_DDF
General	DGOSCGEN	DB2PM_STAT_GENERAL
Buffer Pool	DGOSCBUF	DB2PM_STAT_BUFFER
Group Buffer Pool	DGOSCGBP	DB2PM_STAT_GBUFFER
DDF	DGOSCDDF	DB2PM_STAT_DDF
Dataset	DGOSCSET	DB2PM_STAT_DATASET

Change F to S
for SAVE tables

SMF Data Repository

- Populating PDB
  - Load Cards samples provided KO2SAMP & BBSAMP
  - Load Detail & Summary separately
  - Pre-processing may be needed – Post Midnight & Timezones
  - Load Replace into new partitions

SMF Data Repository

- Database Population
  - RKO2SAMP Dataset

Table	Column Description
DB2PMFACCT_GENERAL	DGOALFGE
DB2PMFACCT_PROGRAM	DGOALFPK
DB2PMFACCT_BUFFER	DGOALFBU
DB2PMFACCT_GBUFFER	DGOALFGP
DB2PMFACCT_DDF	DGOALFDF
DB2PM_STAT_GENERAL	DGOSLGEN
DB2PM_STAT_BUFFER	DGOSLBUF
DB2PM_STAT_GBUFFER	DGOSLGBP
DB2PM_STAT_DDF	DGOSLDDF
DB2PM_STAT_DATASET	DGOSLSET

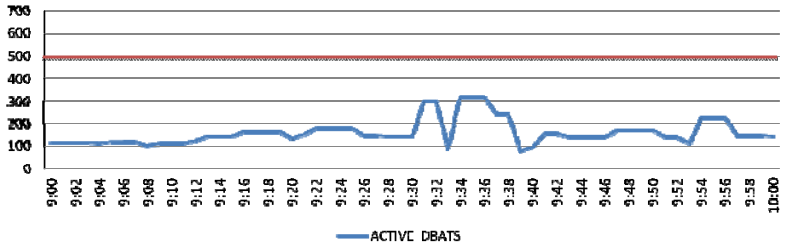
Change F to S
for SAVE tables

SMF Data Repository

- General - Accounting
  - Over 350 Accounting Columns
    - Class 1 & 2 Elapsed/CPU
    - DML & DDL Counters
    - Commit/Rollback
    - RID Pool stats
    - Lock Stats

SMF Data Repository

- General - Statistics
  - Over 650 Counters
    - EDM Pool Stats
    - RID Pool failures
    - MAX Open Datasets
    - Log Reads/Writes
    - 32 different Latch Counters
    - Thread and Connection stats



SMF Data Repository

- Packages
  - Class 7 Elapsed & CPU
  - Class 8 Breakdown – Time & Occurrences
    - Lock/Latch
    - Sync I/O
    - Service Task Suspension
  - zIIP CPU
  - DML Counters

SMF Data Repository

- BPs
  - Statistics Thresholds
    - DWQT/VDWQT/DMHT
    - No Prefetch No Buffer
    - Recall Timeouts/VPOOL Full/Expansion Failed
  - Accounting
    - Getpages
    - Sync Read & Writes
    - Asyn Reads
    - Prefetch
      - Sequential/List/Dynamic

SMF Data Repository

- DDF
  - Statistics
    - Transactions Sent & Received
    - Commits Sent & Received
  - Accounting
    - Includes IP Address
    - DBAT Waiting Time
    - Join to General to capture client info

SMF Data Repository

- Column Descriptions
  - RKO2SAMP Dataset

Table	Column Description
DB2PM_ACCT_GENERAL	DGOABFGE
DB2PM_ACCT_PROGRAM	DGOABFPK
DB2PM_ACCT_BUFFER	DGOABFBU
DB2PM_ACCT_GBUFFER	DGOABFGP
DB2PM_ACCT_DDF	DGOABFDF
DB2PM_STAT_GENERAL	DGOSBGEN
DB2PM_STAT_BUFFER	DGOSBBUF
DB2PM_STAT_GBUFFER	DGOSBGBP
DB2PM_STAT_DDF	DGOSBDDF
DB2PM_STAT_DATASET	DGOSBSET

Change F to S
for SAVE tables

SMF Data Repository

- Column Descriptions – Meta Data in Load Format
- Load Column Meta Data into table
- Example DDL & Data: -

```
CREATE TABLE DSN8610.DB2PM_COLUM_DESCR
    (COLUMN_NAME CHAR(18) NOT NULL,
     FIELD_ID    CHAR(8)  NOT NULL
     DESCRIPTION CHAR(39) NOT NULL,
     SEQ         INTEGER  NOT NULL)
```

```
COLUMN_NAME        FIELD_ID   DESCRIPTION
CLASS8_SYNC_IO     QPACAWTI   The accumulated elapsed wait time for
CLASS8_SYNC_IO     QPACAWTI   I/O suspensions under this thread
CLASS8_SYNC_IO     QPACAWTI   during the execution of the package or
CLASS8_SYNC_IO     QPACAWTI   DBRM.
```

Design

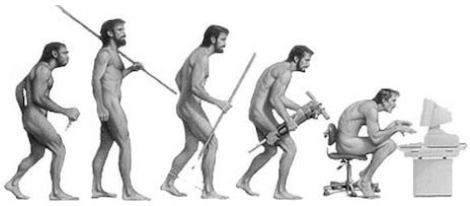
- How can History Help?
  - Introducing PDB
    - What is it…and why do I need one?
- SMF Data Repository
  - Extract
  - Create
  - Populate
- Design (Decisions, Decisions)
  - How Much History
  - Ad-hoc Tables
  - Data Warehouse Location
- Reporting
  - Trend Analysis (predicting the future)
  - Proactive Problem Determination

Design

- Design (Decisions, Decisions)
  - How Much History (Hourly, Daily, Monthly?)
  - Ad-hoc tables
  - PDB Location

Design

- How Much History?
  - Depends on level of change
  - And possibly how much DASD



- Summary Tables
  - Multi Levels of Aggregation, i.e. Daily, Monthly
  - Partitioning
  - Partition Cycle Process
  - Housekeeping

Design

- Ad-hoc Tables
  - One off loads of non-summarised Performance Data
  - Smaller size
  - No need for partitioning
  - Use for Reactive problem determination

Design

- PDB Location
  - Extract & Load process can be CPU hungry
  - Move processing to non-production
  - Don't let your PDB become a production performance issue

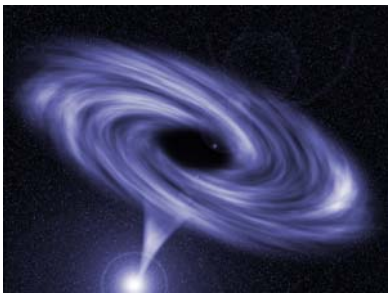
Reporting

- How can history help?
  - Introducing PDB
    - What is it…and why do I need one?
- SMF Data Repository
  - Extract
  - Create
  - Populate
- Design (Decisions, Decisions)
  - How Much History
  - Ad-hoc Tables
  - Data Warehouse Location
- Reporting
  - Trend Analysis (Predicting the Future)
  - Proactive Problem Determination

Reporting

- 35% of Performance Databases are not regularly updated *
- 75% of Performance Databases are not actively reported on *
- So don't let your PDB become a Performance Data Basement

…or even worse a Performance Data Black Hole



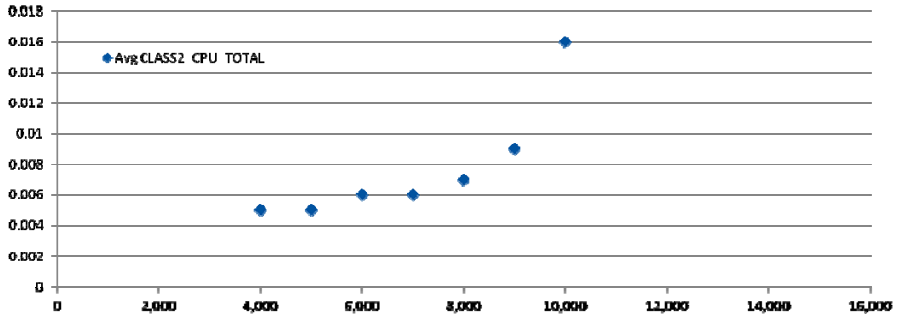
*Over 90% of stats on this slide are made up

Reporting

- Trend Analysis (Predicting the future)
  - Tie up with Application Stats/Business Transaction rates
  - Add bespoke columns of interest
  - Problem Prevented 1
    - Application Workload planned to increase significantly
    - Package level comparison to determine scalability
    - Comparison of CPU consumption with Business Transaction rates
    - Suspect Average CPU time increases at busy times
    - Check Accounting General over differing periods of activity

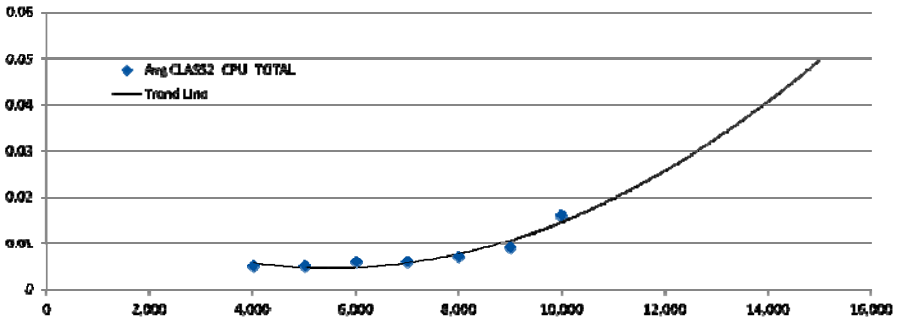
Reporting

- Trend Analysis (Predicting the future)
  - Problem Prevented 1
    - Increase in Class 2 CPU for specific package noted
    - Significant increase as Business Transaction rate increases



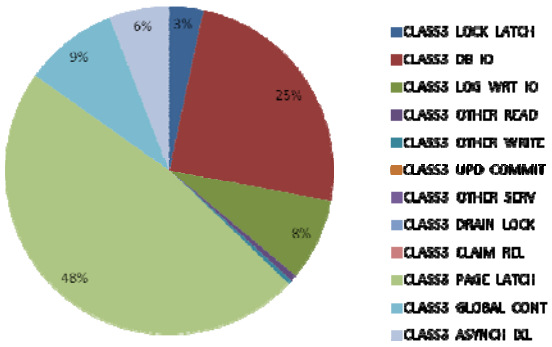
Reporting

- Trend Analysis (Predicting the future)
  - Problem Prevented 1
    - Increase in Class 2 CPU for specific package noted
    - Significant increase as Business Transaction rate increases



Reporting

- Trend Analysis (Predicting the future)
  - Problem Prevented 1
    - Breakdown of CLASS3 time for specific package
    - Majority of time in CLASS3_PAGE_LATCH



Reporting

- Proactive Problem Determination
  - Top 10s - Class 7 Total Elapsed & CPU
  - Also No. of executions to calculate average
  - Top 10 Class 7 CPU consumers - Example

```
SELECT
       DATE(INTERVAL_TIME) AS RUN_DATE
      ,PCK_ID
      ,CLASS7_CPU_AGENT AS C7_CPU
  FROM DB2PMSACCT_PROGRAM
ORDER BY CLASS7_CPU_AGENT DESC
FETCH FIRST 10 ROWS ONLY
;
```

```
  RUN_DATE    PCK_ID   C7_CPU
2013-07-01  Z912_LOG  189.012
2013-07-01  Z402_REC  166.406
2013-07-01  Z901_INI  164.871
2013-07-01  Y906_DIS  149.809
2013-07-01  Z807_INI  145.764

……AND SO ON
```

Reporting

- Proactive Problem Determination
  - Top 10 Class 7 CPU consumers Day by Day - SQL

```
WITH SMF_RANK (C7_CPU_RANK, RUN_DATE, PCK_ID, C7_CPU_TOT) AS
   (SELECT DENSE_RANK() OVER(
                  PARTITION BY INTERVAL_TIME
                  ORDER BY SUM(C7_CPU) DESC) AS C7_CPU_RANK,
           DATE(INTERVAL_TIME),
           PCK_ID,
           SUM(CLASS7_CPU_AGENT) AS C7_CPU_TOT
        FROM DB2PMSACCT_PROGRAM
        GROUP BY DATE(INTERVAL_TIME), PCK_ID
   )
SELECT C7_CPU_RANK,
       MAX(CASE WHEN RUN_DATE = CURRENT DATE - 2 DAYS
           THEN PCK_ID END) AS CURR_DATE_2,
       MAX(CASE WHEN RUN_DATE = CURRENT DATE - 1 DAY
           THEN PCK_ID END) AS CURR_DATE_1,
       MAX(CASE WHEN RUN_DATE = CURRENT DATE
           THEN PCK_ID END) AS CURR_DATE
  FROM SMF_RANK
  GROUP BY C7_CPU_RANK
    HAVING C7_CPU_RANK <= 10
  ORDER BY C7_CPU_RANK
;
```

Reporting

- Proactive Problem Determination
  - Top 10 Class 7 CPU consumers Day by Day - Output

C7_CPU_RANK	CURR_DATE_2	CURR_DATE_1	CURR_DATE
1	Z912_LOG	Z912_LOG	Z912_LOG
2	Z402_REC	Z402_REC	Z901_INI
3	Z901_INI	Z901_INI	Z402_REC
4	Y906_DIS	Z807_INI	Z807_INI
5	Z807_INI	Y906_DIS	Y906_DIS

……AND SO ON

Reporting

- Proactive Problem Determination
  - Problem Prevented 2
    - Release new code into production at low scale to mitigate risk
    - Check daily comparison of Top 10s
    - Top 10 Class 7 Elapsed unchanged
    - But…
    - New entry for Top 10 Class 7 Elapsed Average

Reporting

- Proactive Problem Determination
  - Problem Prevented 2
    - Checked Class 7 Elapsed for Package over last week

INTRVAL_DATE	C7_ELAPSED_AVG	C7_TOTAL
2013-05-31	0.016	565
2013-06-01	0.014	482
2013-06-02	0.014	465
2013-06-03	0.016	581
2013-06-04	0.015	547
2013-06-05	0.015	539
2013-06-06	**0.798**	2763

    - SQL only running on 2 Servers (CLIENT_WSNAME)
    - Inefficient SQL rewritten
    - Full roll out would have resulted in 5000% increase in Class 7 Elapsed

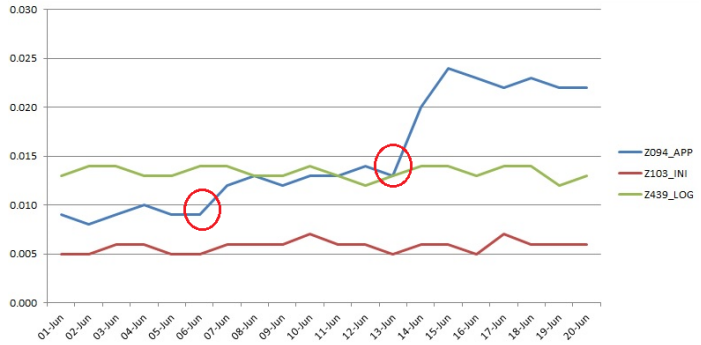
Reporting

- Proactive Problem Determination
  - Problem Prevented 3
    - Significant increase in row volumes across 20 tables
    - Some tables minimally populated
    - Gradual increase in population to mitigate risk
    - Select Average Class 7 Elapsed

```
SELECT
       DATE(A.INTERVAL_TIME) AS RUN_DATE
      ,A.PCK_ID
      ,A.CLASS7_ELAPSED/DECIMAL(A.PCK_RECDS_CLASS7)
            AS C7_ELAP_AVG
  FROM DB2PMSACCT_PROGRAM A
    INNER JOIN
       SYSIBM.SYSPACKDEP B
    ON  A.PCK_ID = B.DNAME
    AND A.PCK_COLLECTION_ID = B.DCOLLID
  WHERE B.BNAME = 'GROWING_TABLE'
    AND B.BQUALIFER = 'PROD'
ORDER BY PCK_ID, RUN_DATE
;
```

Reporting

- Proactive Problem Determination
  - Problem Prevented 3
    - Class 7 Elapsed Average by Package
    - Data population on 6th June & 13th June



Recap

- Why a PDB?
  - Prevent a performance disaster before it happens
  - Proactive monitoring & trend analysis
- Design
  - Plan ahead. Don't let PDB become biggest consumer of CPU.
- SMF Data Repository
  - Refer to documentation & Samples. Will save months of effort
- Reports
  - Don't leave your PDW idle and become a forgotten Data Basement
  - Or even worse a CPU consuming, DASD hungry Data Black Hole

“Moltes gràcies pel seu temps”

Paul Stoker
Triton Consulting
paul.stoker@triton.co.uk

F05
Performance Perils Prevented
– How History Helps

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988