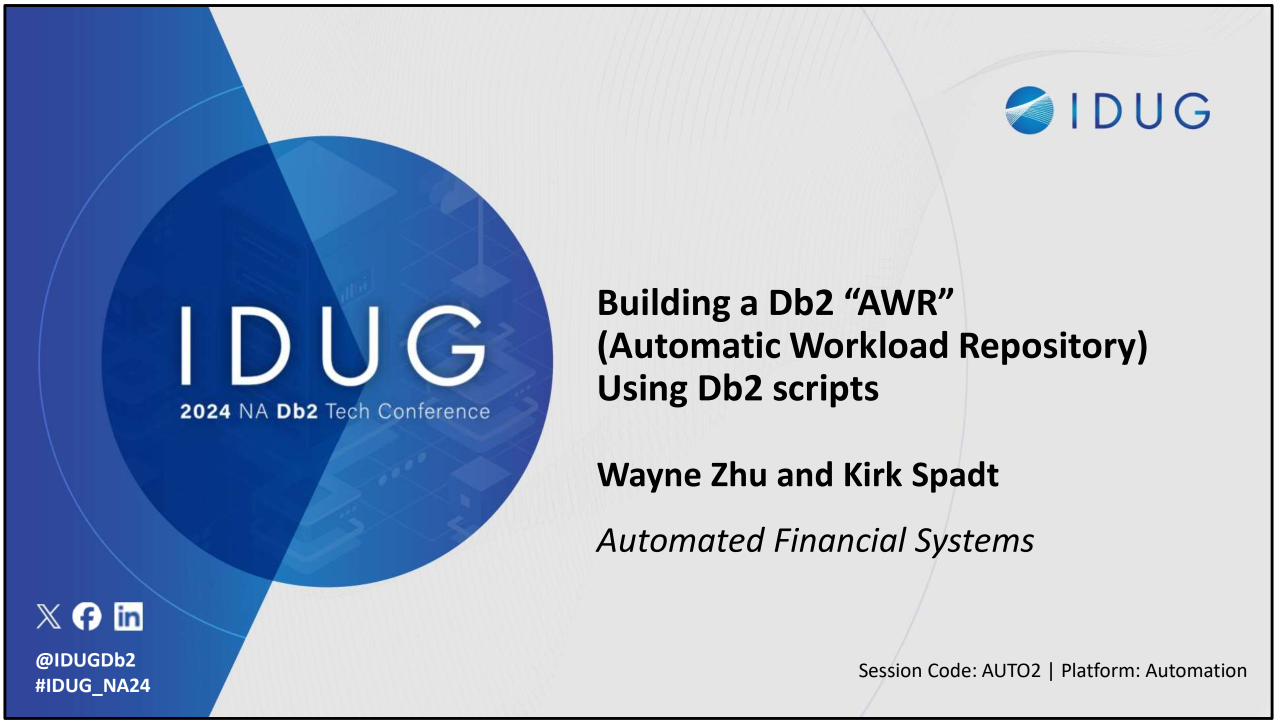

IDUG
IDUG
2024 NA Db2 Tech Conference

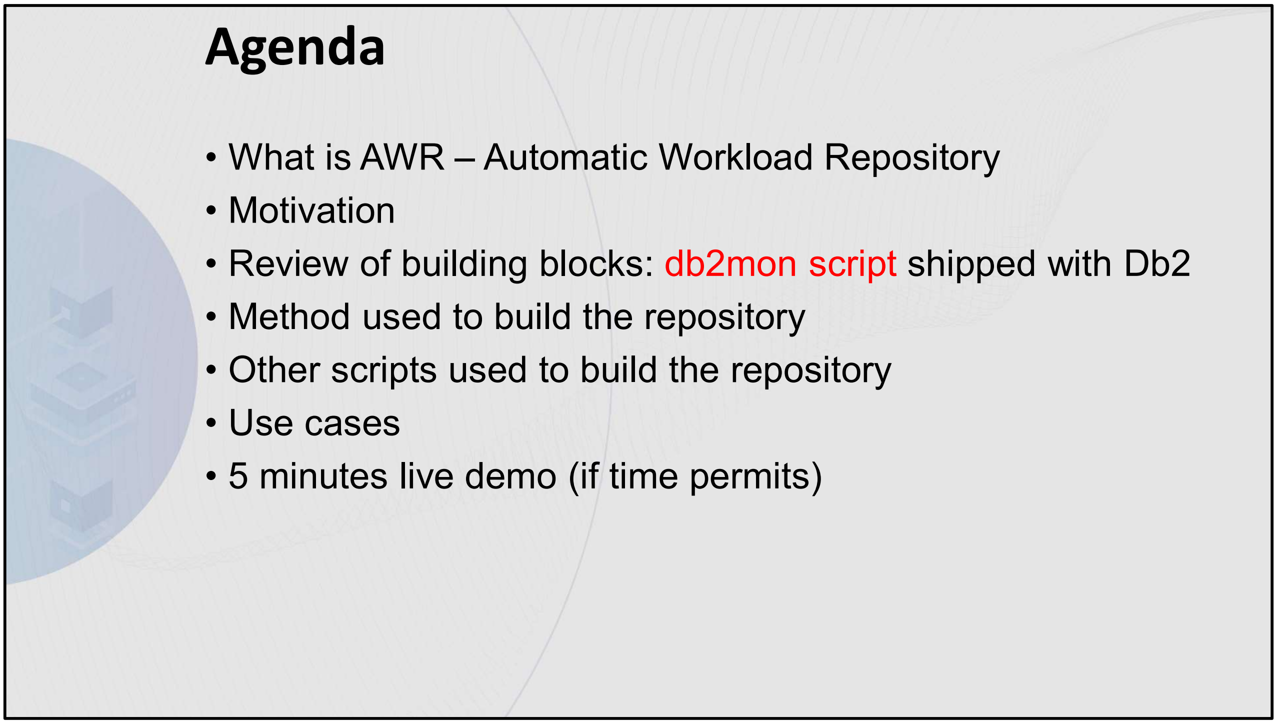
Building a Db2 “AWR” (Automatic Workload Repository) Using Db2 scripts

Wayne Zhu and Kirk Spadt

Automated Financial Systems

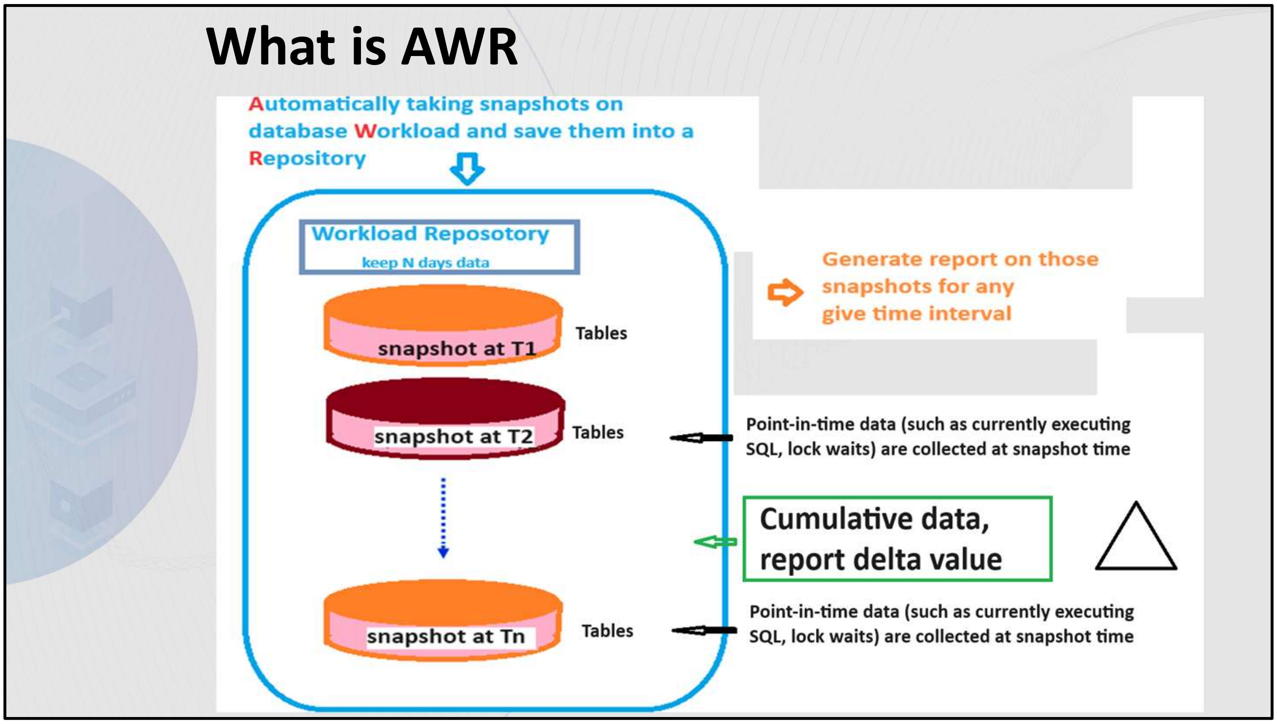
@IDUGDb2
#IDUG_NA24

Session Code: AUTO2 | Platform: Automation



Agenda

- What is AWR – Automatic Workload Repository
- Motivation
- Review of building blocks: db2mon script shipped with Db2
- Method used to build the repository
- Other scripts used to build the repository
- Use cases
- 5 minutes live demo (if time permits)



What is AWR

Automatically taking snapshots on database Workload and save them into a Repository

Workload Reposotory

keep N days data

snapshot at T1 Tables

snapshot at T2 Tables

snapshot at Tn Tables

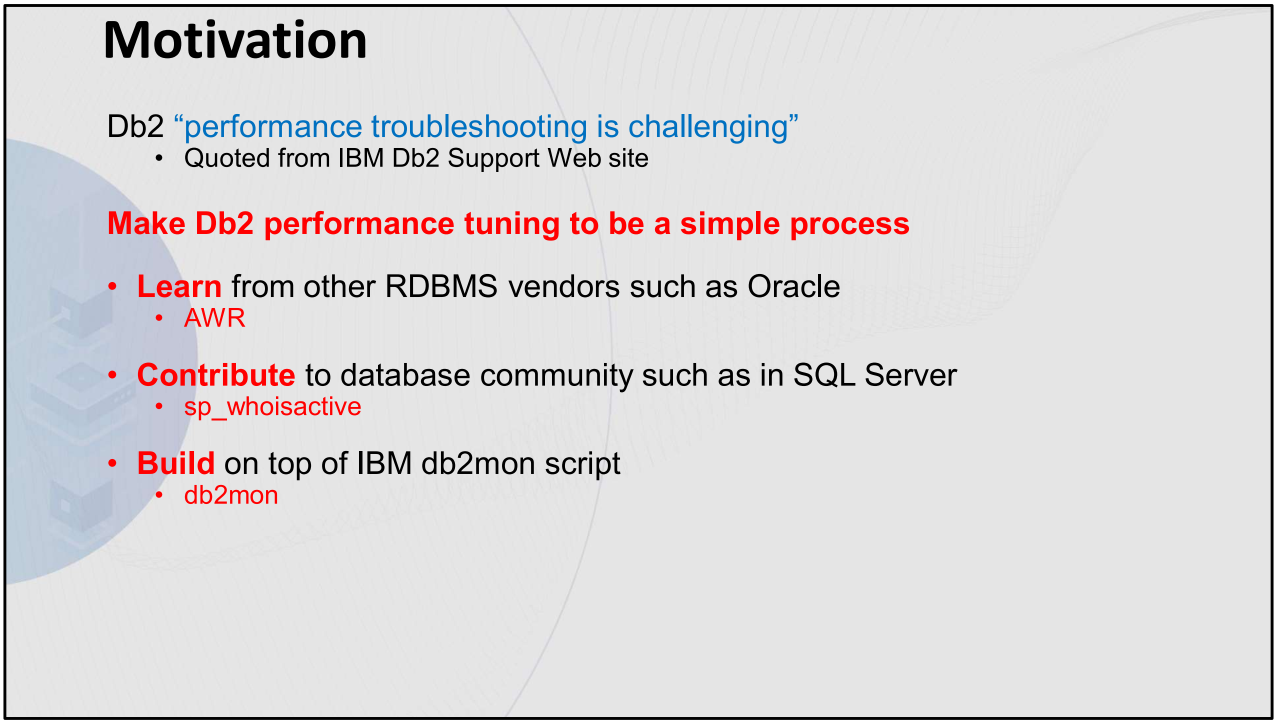
Generate report on those snapshots for any give time interval

Point-in-time data (such as currently executing SQL, lock waits) are collected at snapshot time

Cumulative data, report delta value △

Point-in-time data (such as currently executing SQL, lock waits) are collected at snapshot time

Report on PIT data and Delta data



Motivation

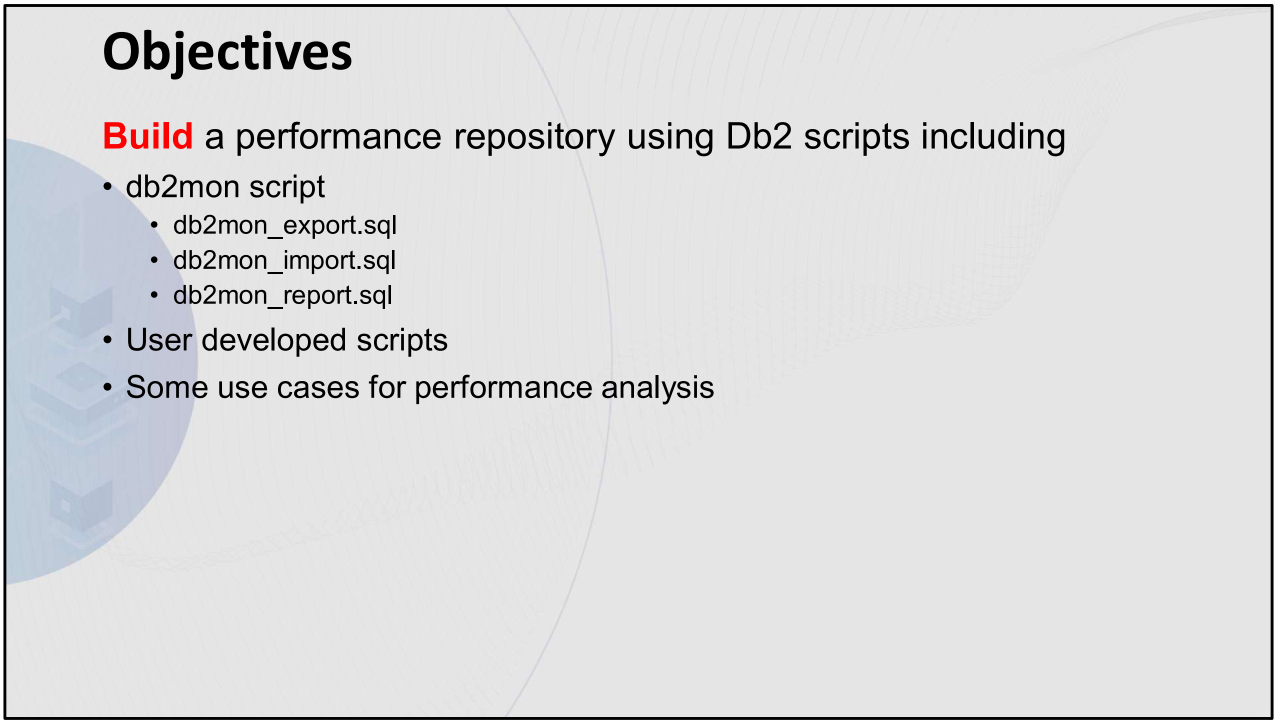
Db2 “performance troubleshooting is challenging”

- Quoted from IBM Db2 Support Web site

Make Db2 performance tuning to be a simple process

- **Learn** from other RDBMS vendors such as Oracle
  - AWR
- **Contribute** to database community such as in SQL Server
  - sp_whoisactive
- **Build** on top of IBM db2mon script
  - db2mon

Db2 performance tuning made simple
Learn from Others
Contribute
Leverage Db2 technolgy



Objectives

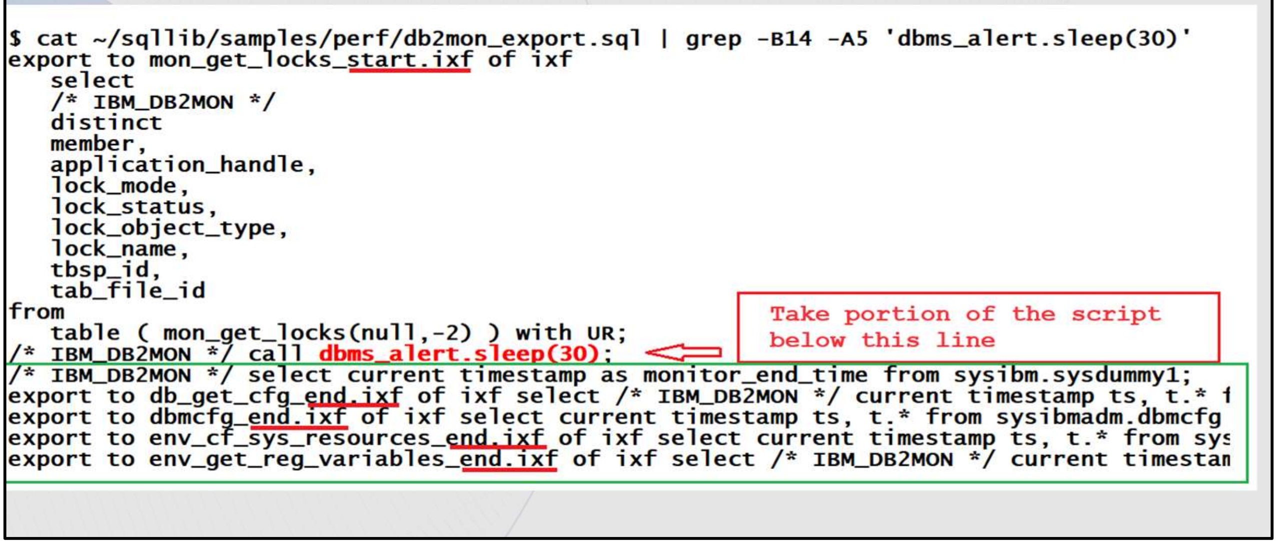
Build a performance repository using Db2 scripts including

- db2mon script
  - db2mon_export.sql
  - db2mon_import.sql
  - db2mon_report.sql
- User developed scripts
- Some use cases for performance analysis

Build Export Import Report Save and use cases

db2mon script: db2mon_export.sql

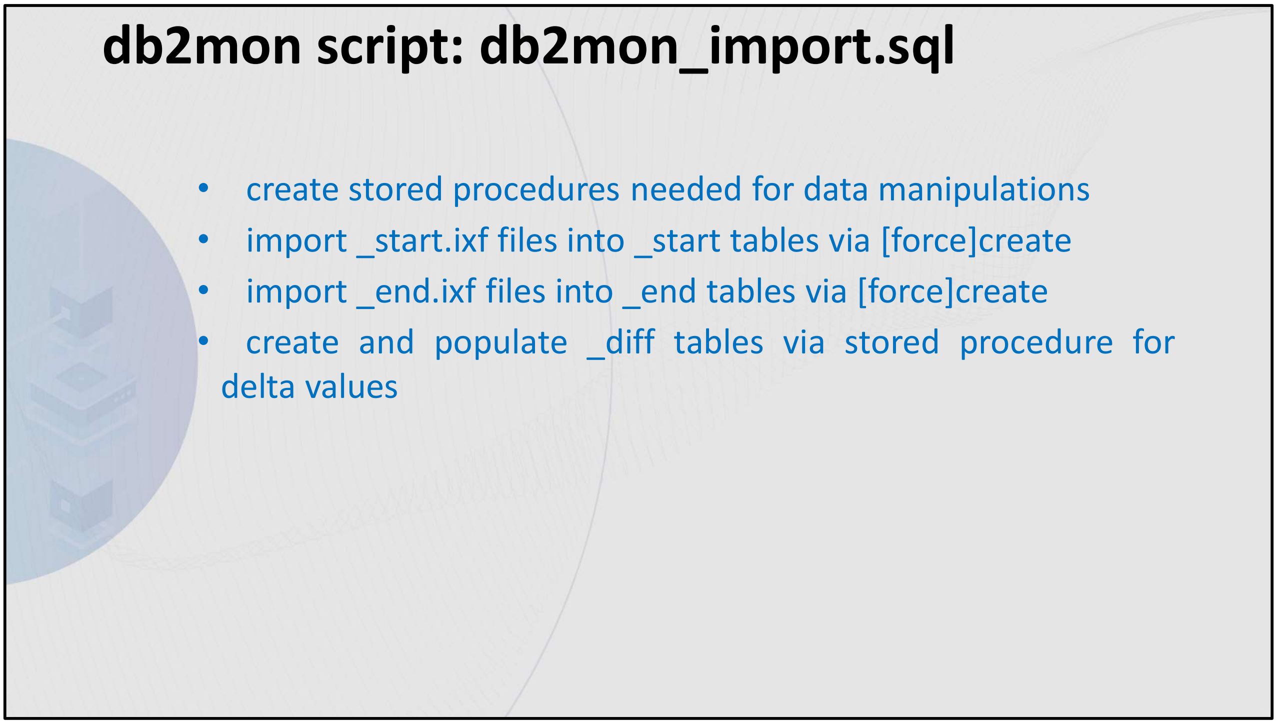
- Export a set of files ending with **_start.ixf**; wait 30 seconds; export another set of files ending **_end.ixf**
- We use part of the script after the line of "sleep (30)" as shown below:



```
$ cat ~/sqllib/samples/perf/db2mon_export.sql | grep -B14 -A5 'dbms_alert.sleep(30)'
export to mon_get_locks_start.ixf of ixf
   select
   /* IBM_DB2MON */
   distinct
   member,
   application_handle,
   lock_mode,
   lock_status,
   lock_object_type,
   lock_name,
   tbsp_id,
   tab_file_id
from
   table ( mon_get_locks(null,-2) ) with UR;
/* IBM_DB2MON */ call dbms_alert.sleep(30);
/* IBM_DB2MON */ select current timestamp as monitor_end_time from sysibm.sysdummy1;
export to db_get_cfg_end.ixf of ixf select /* IBM_DB2MON */ current timestamp ts, t.* f
export to dbmcfg_end.ixf of ixf select current timestamp ts, t.* from sysibmadm.dbmcfg
export to env_cf_sys_resources_end.ixf of ixf select current timestamp ts, t.* from sys
export to env_get_reg_variables_end.ixf of ixf select /* IBM_DB2MON */ current timestan
```

Take portion of the script below this line

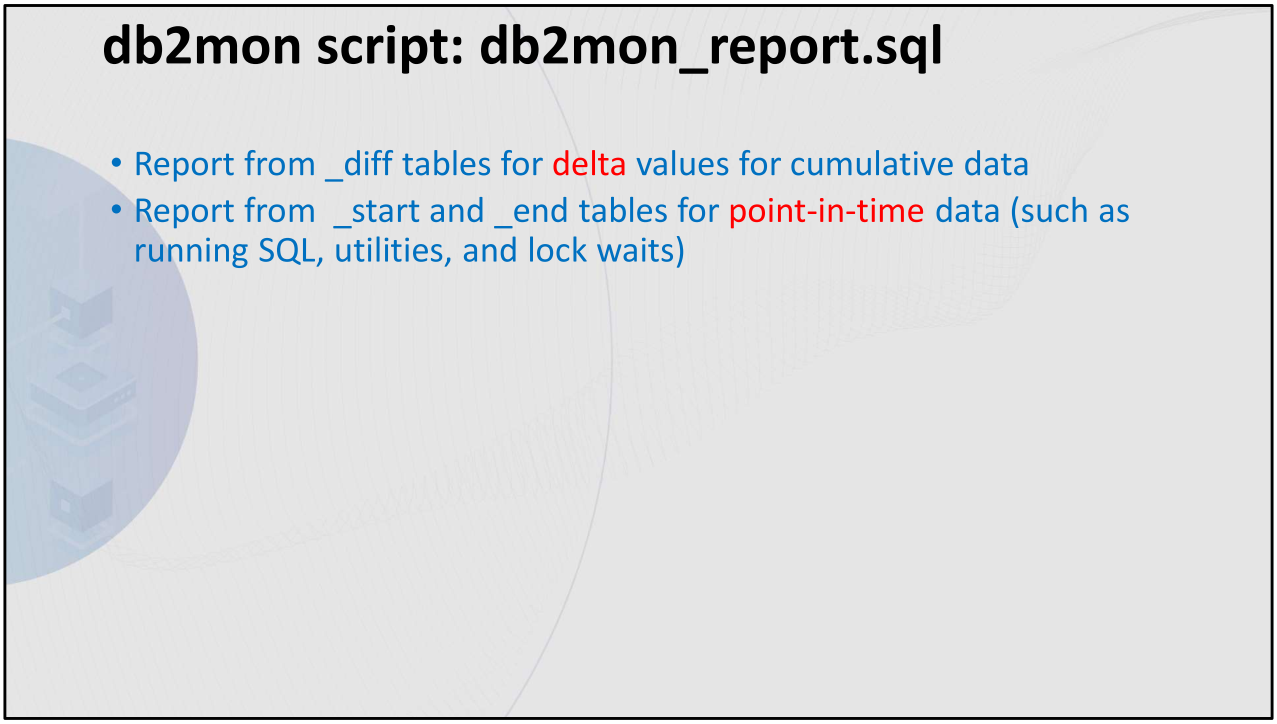
Location ~/sqllib/samples/perf



db2mon script: db2mon_import.sql

- create stored procedures needed for data manipulations
- import _start.ixf files into _start tables via [force]create
- import _end.ixf files into _end tables via [force]create
- create and populate _diff tables via stored procedure for delta values

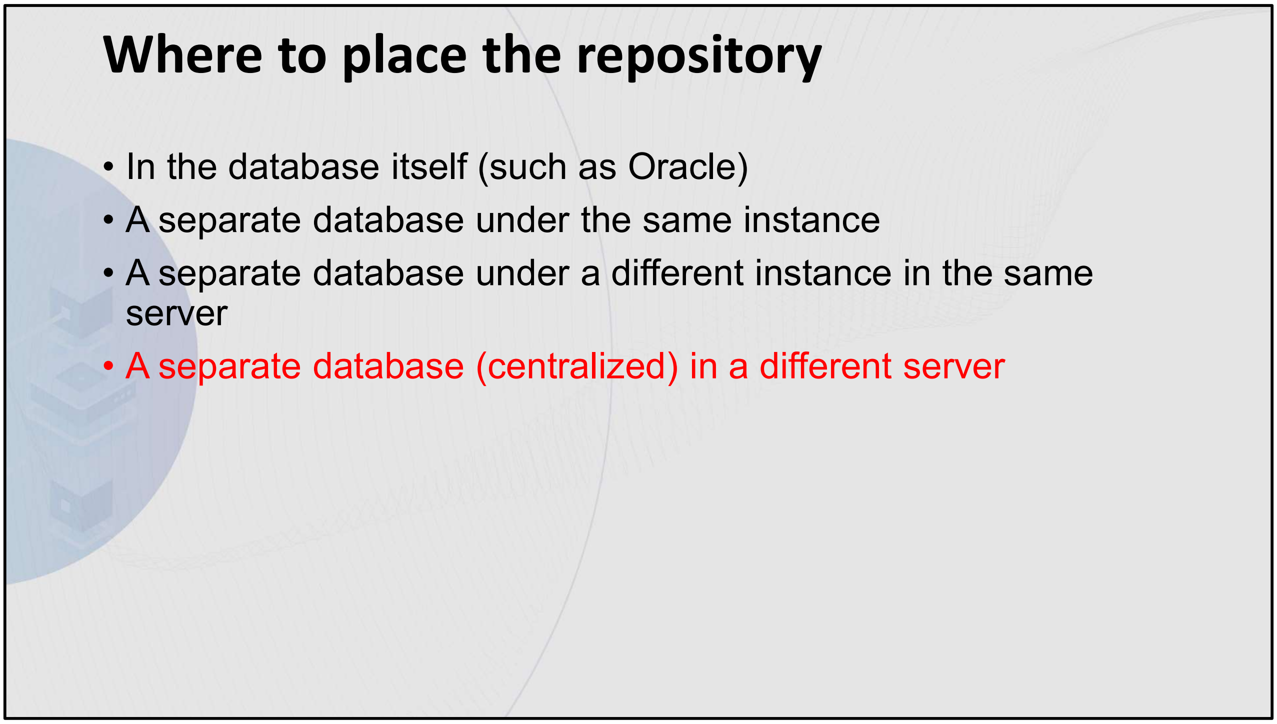
This slide shows the details of the IBM scripts.
Create SP Import IXF start and end   Finally diff the deltas



db2mon script: db2mon_report.sql

- Report from _diff tables for delta values for cumulative data
- Report from _start and _end tables for point-in-time data (such as running SQL, utilities, and lock waits)

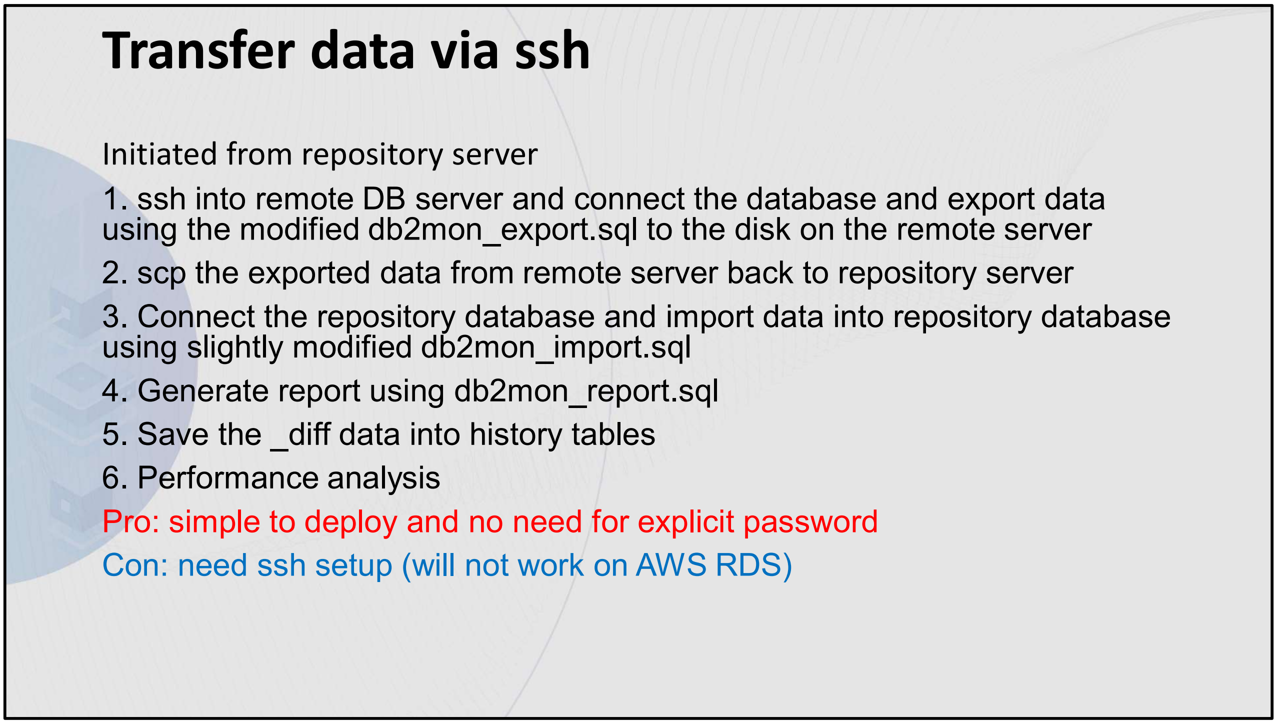
This slide shows the details of the IBM scripts.
Create SP Import IXF start and end   Finally diff the deltas



Where to place the repository

- In the database itself (such as Oracle)
- A separate database under the same instance
- A separate database under a different instance in the same server
- A separate database (centralized) in a different server

You have 4 options of where to place the repo
DB, separate DB separate instance and separate centralized server which is out preference.
1. Reporting load off main server
To compare across the whole farm using SCHEMA to qualify



Transfer data via ssh

Initiated from repository server

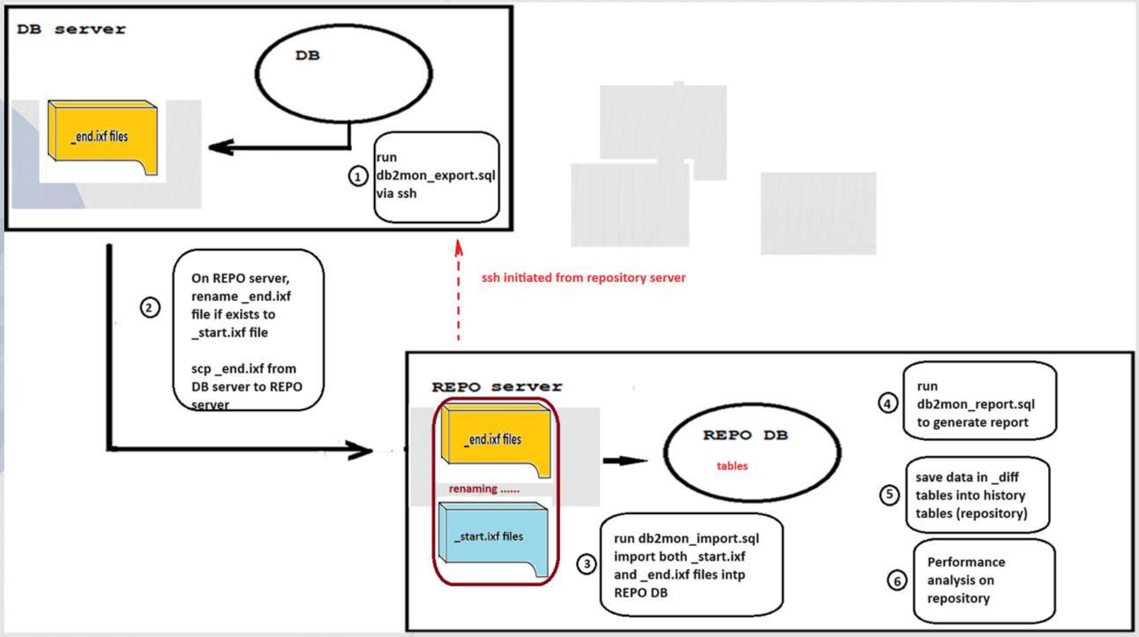
1. ssh into remote DB server and connect the database and export data using the modified db2mon_export.sql to the disk on the remote server
2. scp the exported data from remote server back to repository server
3. Connect the repository database and import data into repository database using slightly modified db2mon_import.sql
4. Generate report using db2mon_report.sql
5. Save the _diff data into history tables
6. Performance analysis

Pro: simple to deploy and no need for explicit password

Con: need ssh setup (will not work on AWS RDS)

This oulines Passwordless SSH method to transfer data to local disk
Other methods are Load from cursor, export, federation

Transfer data via ssh



1 run db2mon_export.sql via ssh

2 On REPO server, rename _end.ixf file if exists to _start.ixf file

scp _end.ixf from DB server to REPO server

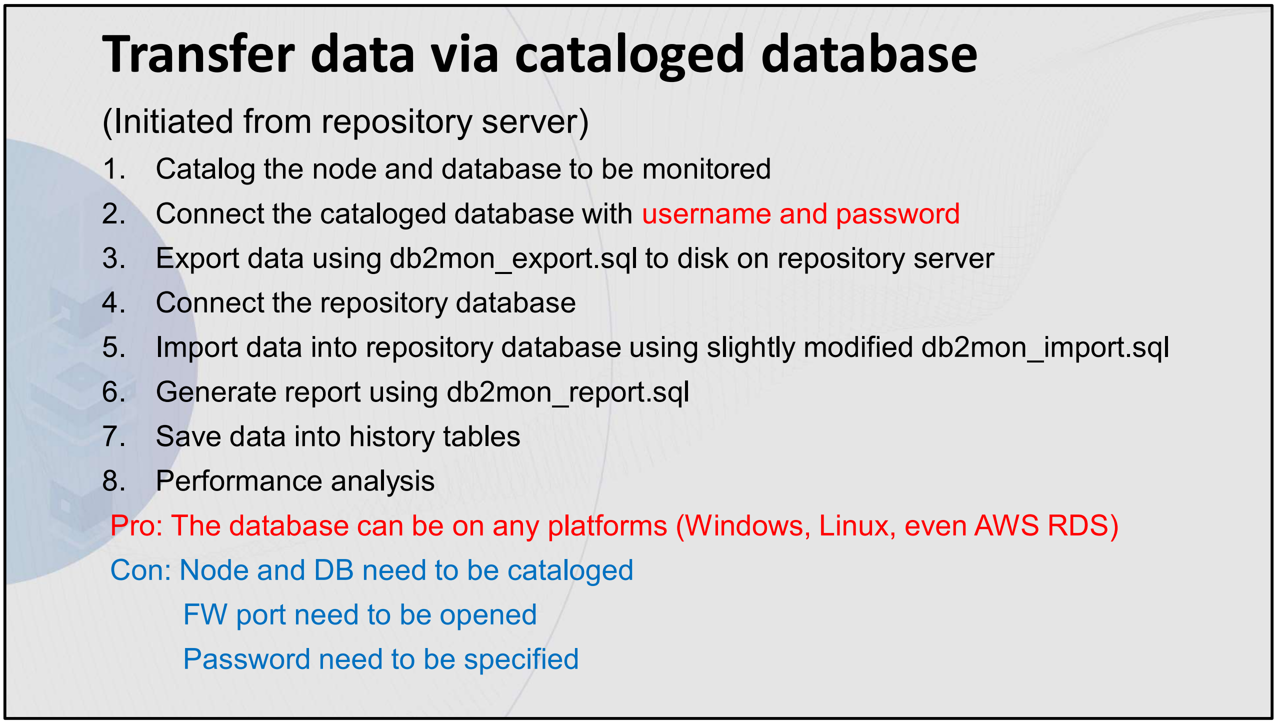
3 run db2mon_import.sql import both _start.ixf and _end.ixf files intp REPO DB

4 run db2mon_report.sql to generate report

5 save data in _diff tables into history tables (repository)

6 Performance analysis on repository

Add # 6
“run export via db2mon_export.sql from the REPO server’
Run scp of the IXF file from the REPO server
NOTE Use the same sizes
Use ther rectalge tiool



Transfer data via cataloged database

(Initiated from repository server)

1. Catalog the node and database to be monitored
2. Connect the cataloged database with username and password
3. Export data using db2mon_export.sql to disk on repository server
4. Connect the repository database
5. Import data into repository database using slightly modified db2mon_import.sql
6. Generate report using db2mon_report.sql
7. Save data into history tables
8. Performance analysis

Pro: The database can be on any platforms (Windows, Linux, even AWS RDS)

Con: Node and DB need to be cataloged
FW port need to be opened
Password need to be specified

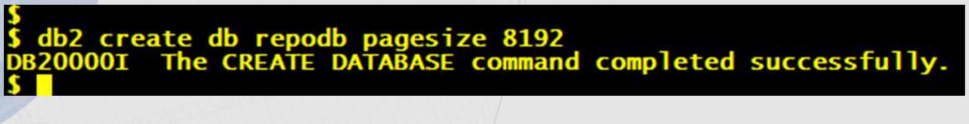
Outlines a Generic method to transfer data to local disk

Build the repository: create the REPO DB

- The repo database is created using 8k pages:

  db2 create database <dbname> pagesize 8192

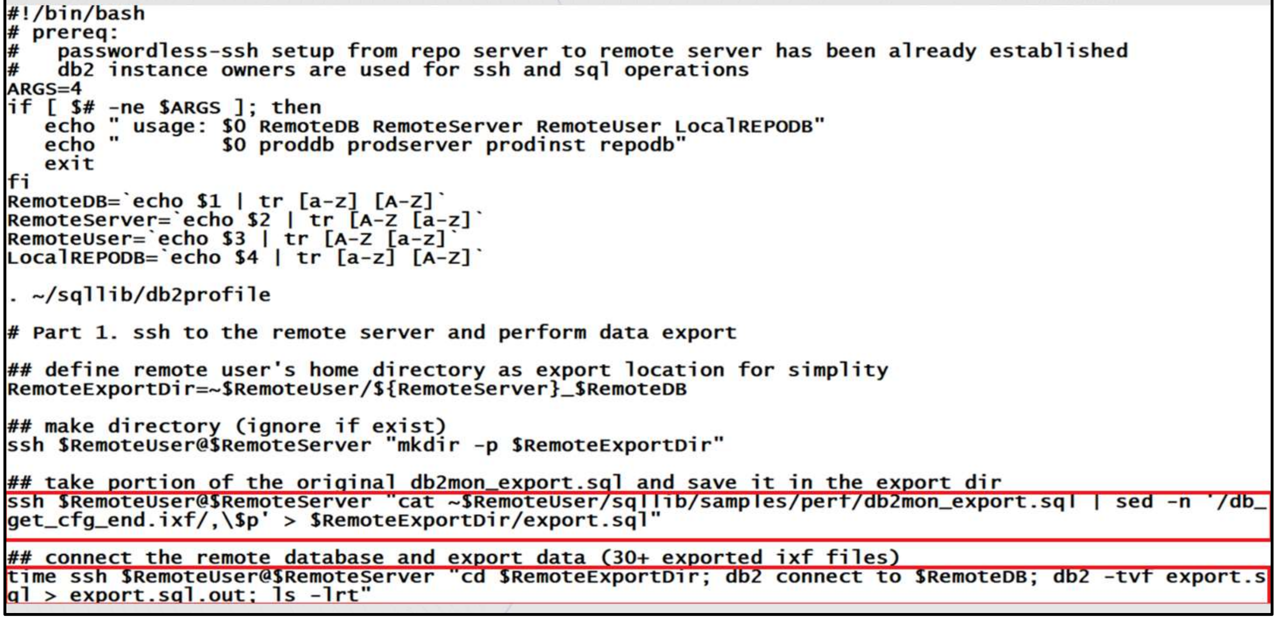
  For example:



```
$
$ db2 create db repodb pagesize 8192
DB20000I  The CREATE DATABASE command completed successfully.
$
```

Build the repository: script part 1 – export data

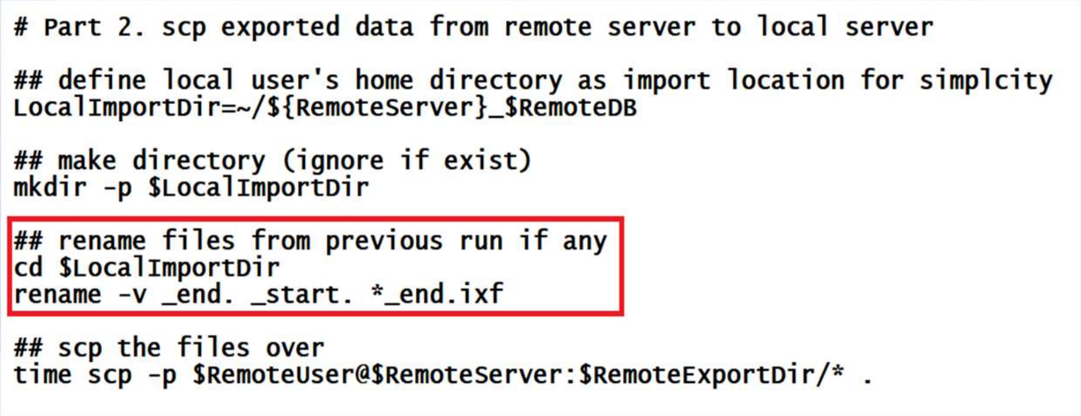
On REBO server, ssh to DB server, take portion of db2mon_export.sql and perform data export



```
#!/bin/bash
# prereq:
#   passwordless-ssh setup from repo server to remote server has been already established
#   db2 instance owners are used for ssh and sql operations
ARGS=4
if [ $# -ne $ARGS ]; then
   echo " usage: $0 RemoteDB RemoteServer RemoteUser LocalREPODB"
   echo "        $0 proddb prodserver prodinst repodb"
   exit
fi
RemoteDB=`echo $1 | tr [a-z] [A-Z]`
RemoteServer=`echo $2 | tr [A-Z [a-z]`
RemoteUser=`echo $3 | tr [A-Z [a-z]`
LocalREPODB=`echo $4 | tr [a-z] [A-Z]`

. ~/sqllib/db2profile

# Part 1. ssh to the remote server and perform data export

## define remote user's home directory as export location for simplity
RemoteExportDir=~$RemoteUser/${RemoteServer}_$RemoteDB

## make directory (ignore if exist)
ssh $RemoteUser@$RemoteServer "mkdir -p $RemoteExportDir"

## take portion of the original db2mon_export.sql and save it in the export dir
ssh $RemoteUser@$RemoteServer "cat ~$RemoteUser/sqllib/samples/perf/db2mon_export.sql | sed -n '/db_
get_cfg_end.ixf/,\$p' > $RemoteExportDir/export.sql"

## connect the remote database and export data (30+ exported ixf files)
time ssh $RemoteUser@$RemoteServer "cd $RemoteExportDir; db2 connect to $RemoteDB; db2 -tvf export.s
ql > export.sql.out; ls -lrt"
```

Build the repository: Script Part 2 – scp the data

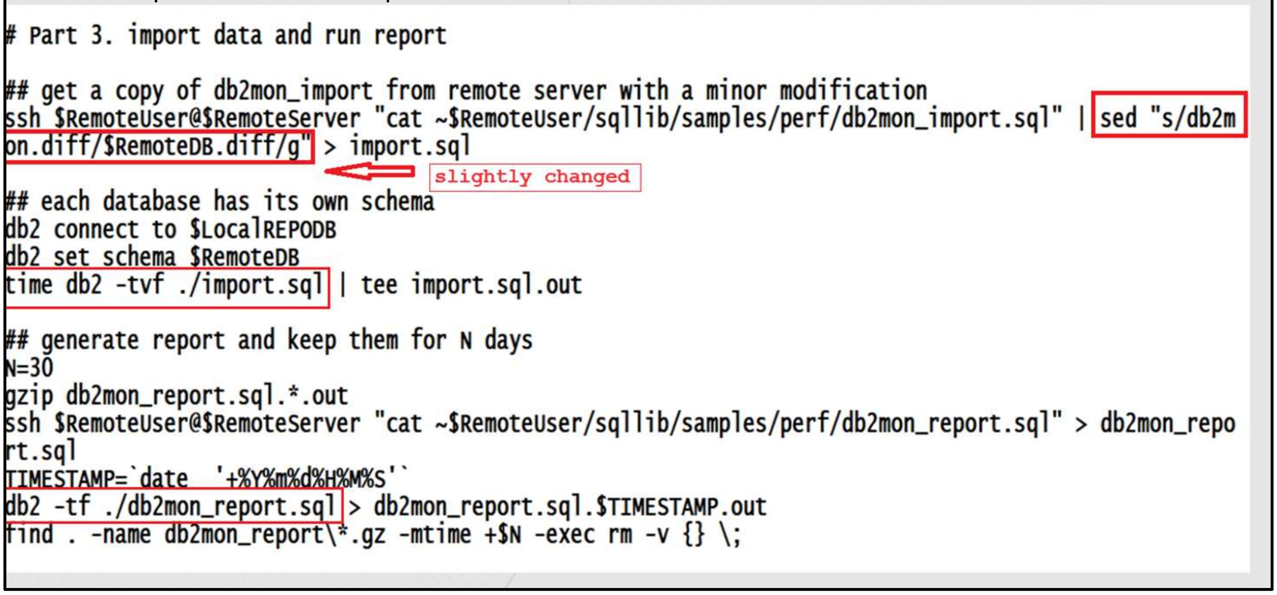
1) On repositorty server, rename _enf.ixf to _start.ixf files if any
2) scp the exported _end.ixf files from remote server to repository server



```
# Part 2. scp exported data from remote server to local server

## define local user's home directory as import location for simplcity
LocalImportDir=~/${RemoteServer}_$RemoteDB

## make directory (ignore if exist)
mkdir -p $LocalImportDir

## rename files from previous run if any
cd $LocalImportDir
rename -v _end. _start. *_end.ixf

## scp the files over
time scp -p $RemoteUser@$RemoteServer:$RemoteExportDir/* .
```

Build the repository: Script Part 3 - import and report

- import data and run report

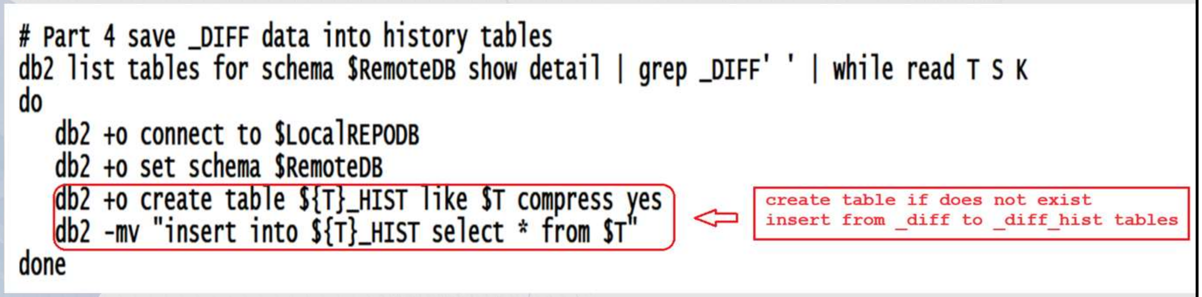


```
# Part 3. import data and run report

## get a copy of db2mon_import from remote server with a minor modification
ssh $RemoteUser@$RemoteServer "cat ~$RemoteUser/sqllib/samples/perf/db2mon_import.sql" | sed "s/db2m
on.diff/$RemoteDB.diff/g" > import.sql

## each database has its own schema
db2 connect to $LocalREPODB
db2 set schema $RemoteDB
time db2 -tvf ./import.sql | tee import.sql.out

## generate report and keep them for N days
N=30
gzip db2mon_report.sql.*.out
ssh $RemoteUser@$RemoteServer "cat ~$RemoteUser/sqllib/samples/perf/db2mon_report.sql" > db2mon_repo
rt.sql
TIMESTAMP=`date  '+%Y%m%d%H%M%S'`
db2 -tf ./db2mon_report.sql > db2mon_report.sql.$TIMESTAMP.out
find . -name db2mon_report\*.gz -mtime +$N -exec rm -v {} \;
```

slightly changed

Build the repository: Script Part 4 - save to history

- Save _diff data into history tables

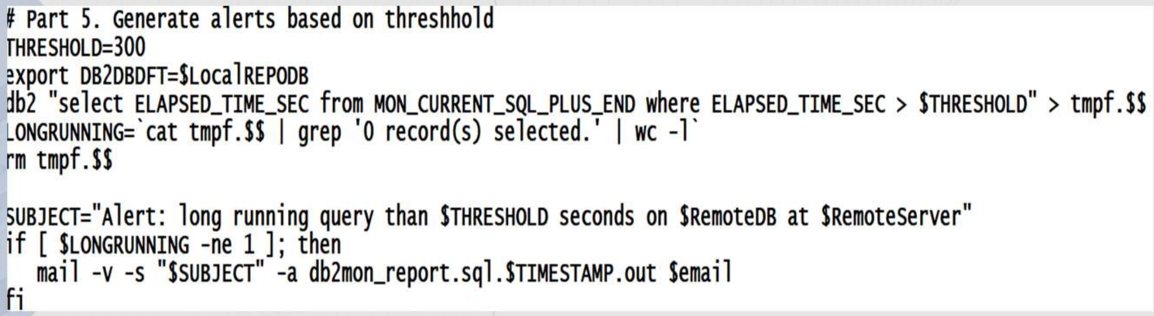


```
# Part 4 save _DIFF data into history tables
db2 list tables for schema $RemoteDB show detail | grep _DIFF' ' | while read T S K
do
   db2 +o connect to $LocalREPODB
   db2 +o set schema $RemoteDB
   db2 +o create table ${T}_HIST like $T compress yes
   db2 -mv "insert into ${T}_HIST select * from $T"
done
```

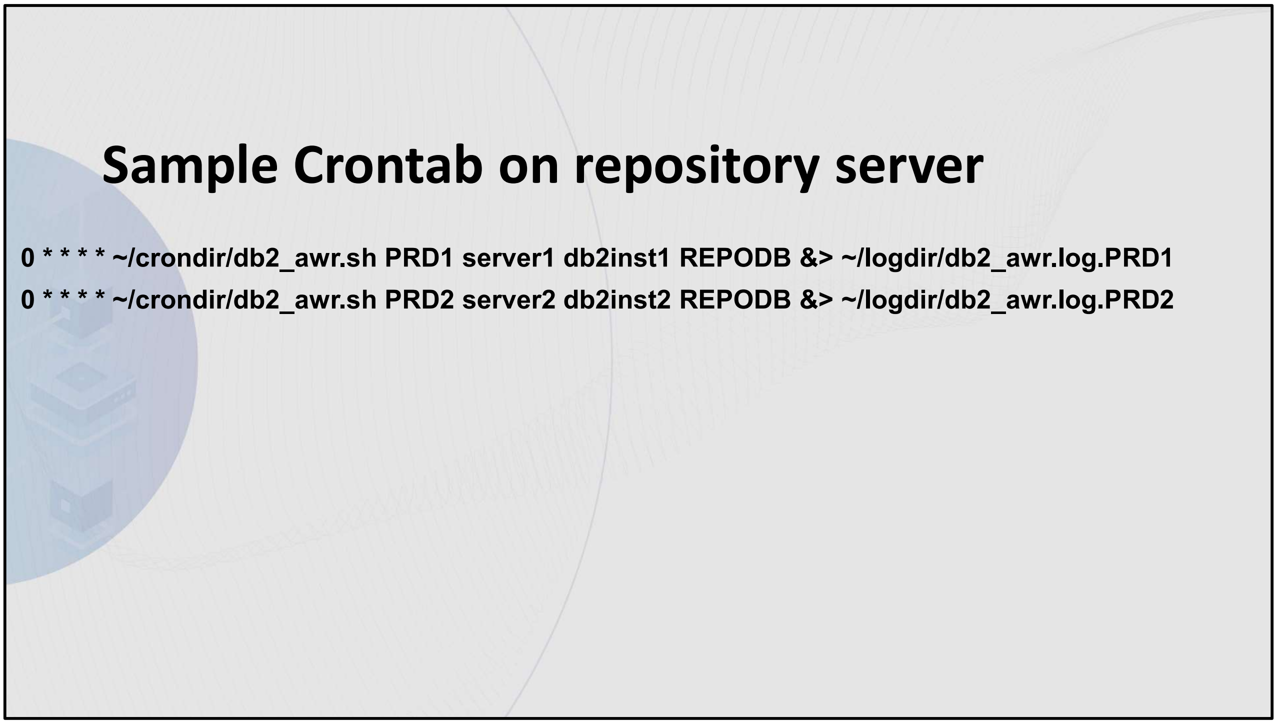
create table if does not exist
insert from _diff to _diff_hist tables

Build the repository: Script Part 5 – alerts

- Monitoring long running SQL

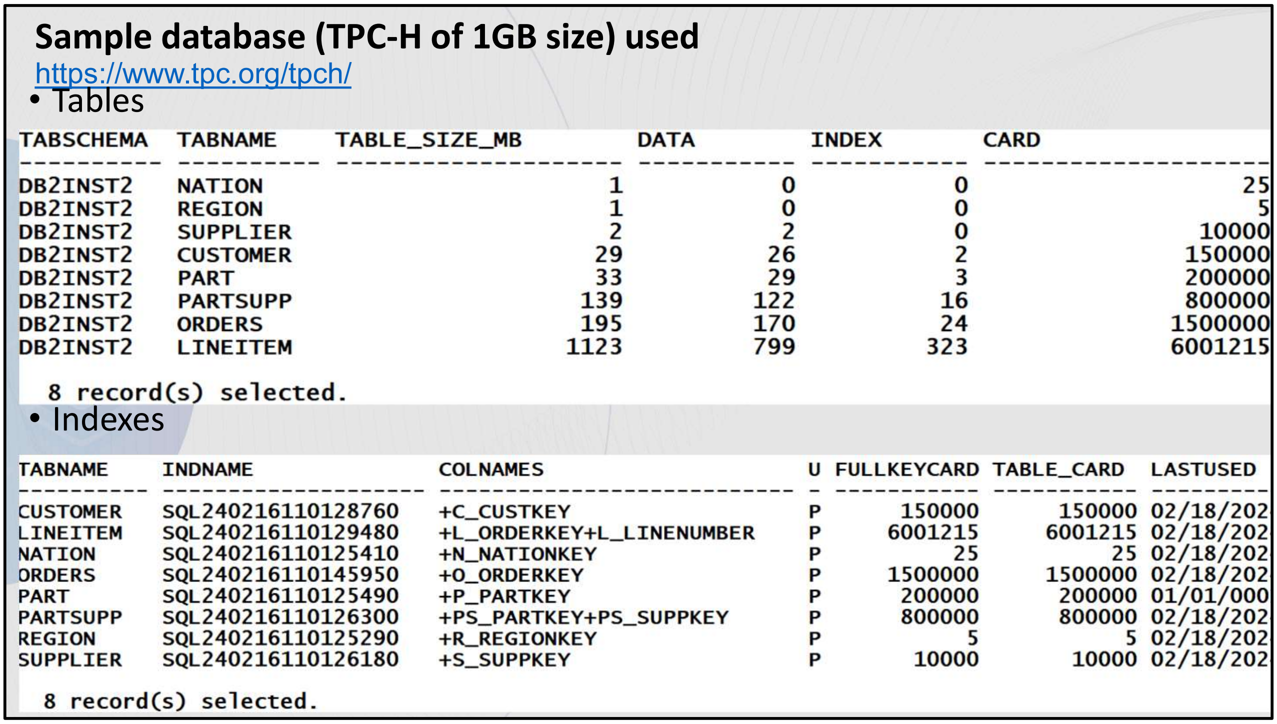


```
# Part 5. Generate alerts based on threshhold
THRESHOLD=300
export DB2DBDFT=$LocalREPODB
db2 "select ELAPSED_TIME_SEC from MON_CURRENT_SQL_PLUS_END where ELAPSED_TIME_SEC > $THRESHOLD" > tmpf.$$
LONGRUNNING=`cat tmpf.$$ | grep '0 record(s) selected.' | wc -l`
rm tmpf.$$

SUBJECT="Alert: long running query than $THRESHOLD seconds on $RemoteDB at $RemoteServer"
if [ $LONGRUNNING -ne 1 ]; then
   mail -v -s "$SUBJECT" -a db2mon_report.sql.$TIMESTAMP.out $email
fi
```



Sample Crontab on repository server

```
0 * * * * ~/crondir/db2_awr.sh PRD1 server1 db2inst1 REPODB &> ~/logdir/db2_awr.log.PRD1
0 * * * * ~/crondir/db2_awr.sh PRD2 server2 db2inst2 REPODB &> ~/logdir/db2_awr.log.PRD2
```



Sample database (TPC-H of 1GB size) used

https://www.tpc.org/tpch/

- Tables

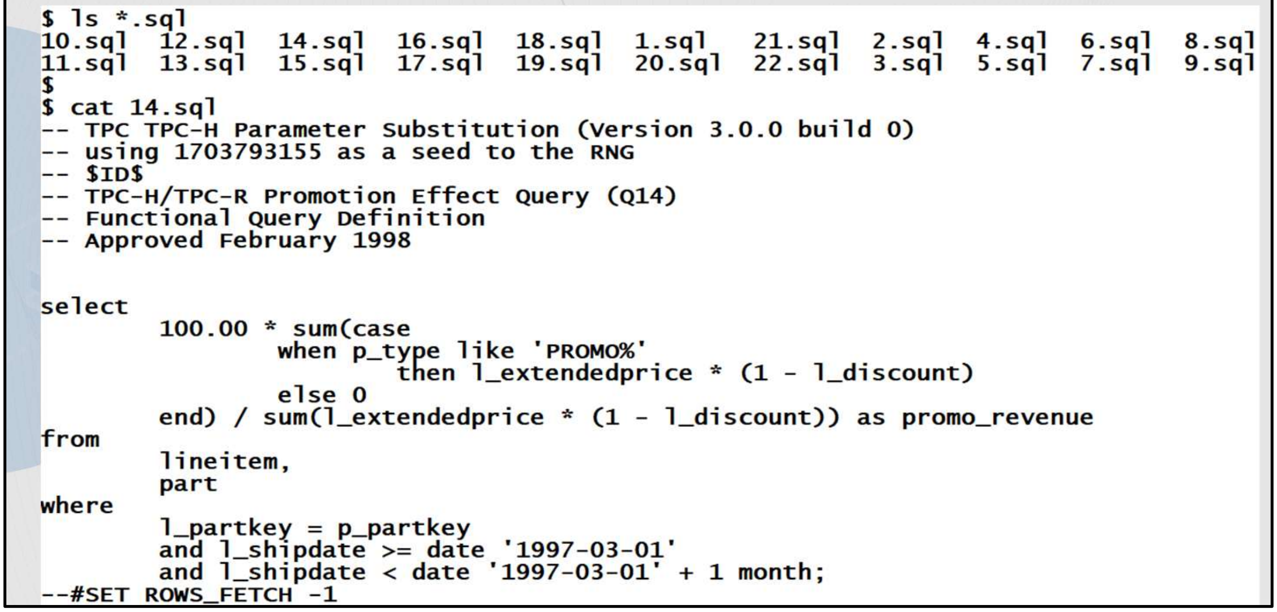
```
TABSCHEMA  TABNAME    TABLE_SIZE_MB        DATA        INDEX       CARD
---------- ---------- -------------------- ----------- ----------- --------------------
DB2INST2   NATION                        1           0           0                   25
DB2INST2   REGION                        1           0           0                    5
DB2INST2   SUPPLIER                      2           2           0                10000
DB2INST2   CUSTOMER                     29          26           2               150000
DB2INST2   PART                         33          29           3               200000
DB2INST2   PARTSUPP                    139         122          16               800000
DB2INST2   ORDERS                      195         170          24              1500000
DB2INST2   LINEITEM                   1123         799         323              6001215

  8 record(s) selected.
```

- Indexes

```
TABNAME    INDNAME              COLNAMES                    U FULLKEYCARD TABLE_CARD  LASTUSED
---------- -------------------- --------------------------- - ----------- ----------- ---------
CUSTOMER   SQL240216110128760   +C_CUSTKEY                  P      150000      150000 02/18/202
LINEITEM   SQL240216110129480   +L_ORDERKEY+L_LINENUMBER    P     6001215     6001215 02/18/202
NATION     SQL240216110125410   +N_NATIONKEY                P          25          25 02/18/202
ORDERS     SQL240216110145950   +O_ORDERKEY                 P     1500000     1500000 02/18/202
PART       SQL240216110125490   +P_PARTKEY                  P      200000      200000 01/01/000
PARTSUPP   SQL240216110126300   +PS_PARTKEY+PS_SUPPKEY      P      800000      800000 02/18/202
REGION     SQL240216110125290   +R_REGIONKEY                P           5           5 02/18/202
SUPPLIER   SQL240216110126180   +S_SUPPKEY                  P       10000       10000 02/18/202

  8 record(s) selected.
```

Queries to generate workload

- TPC-H example query:



```
$ ls *.sql
10.sql  12.sql  14.sql  16.sql  18.sql  1.sql   21.sql  2.sql  4.sql  6.sql  8.sql
11.sql  13.sql  15.sql  17.sql  19.sql  20.sql  22.sql  3.sql  5.sql  7.sql  9.sql
$
$ cat 14.sql
-- TPC TPC-H Parameter Substitution (Version 3.0.0 build 0)
-- using 1703793155 as a seed to the RNG
-- $ID$
-- TPC-H/TPC-R Promotion Effect Query (Q14)
-- Functional Query Definition
-- Approved February 1998


select
        100.00 * sum(case
                when p_type like 'PROMO%'
                        then l_extendedprice * (1 - l_discount)
                else 0
        end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
        lineitem,
        part
where
        l_partkey = p_partkey
        and l_shipdate >= date '1997-03-01'
        and l_shipdate < date '1997-03-01' + 1 month;
--#SET ROWS_FETCH -1
```

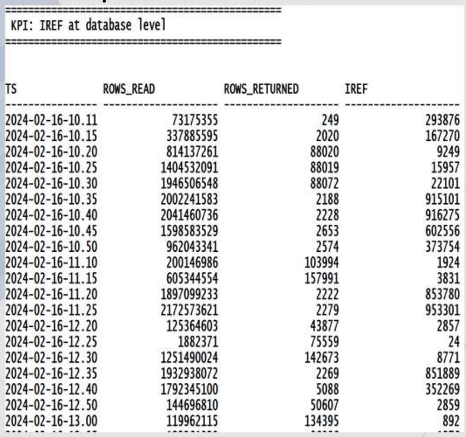
Use case # 1 Establish baseline per hour/per day (trending)

https://www.dbisoftware.com/blog/db2_performance.php?id=95 **“DB2 LUW Performance: Index Read Efficiency (IREF)”**

IREF = Rows read / Rows Selected (Fetched)

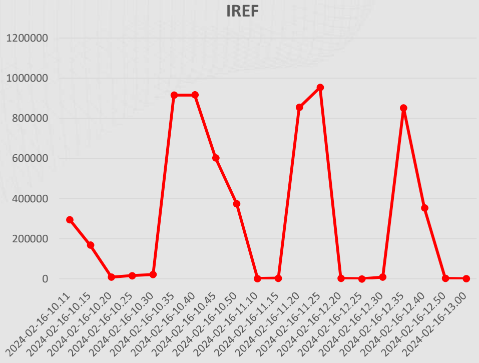
For OLTP:	**IREF < 10**	**Excellent**
	IREF > 10	**Fair**
	IREF > 100	**Poor**
	IREF > 1000	**??????**

- Example data



```
=============================================
 KPI: IREF at database level
=============================================


TS               ROWS_READ            ROWS_RETURNED        IREF
---------------- -------------------- -------------------- --------------------
2024-02-16-10.11             73175355                  249               293876
2024-02-16-10.15            337885595                 2020               167270
2024-02-16-10.20            814137261                88020                 9249
2024-02-16-10.25           1404532091                88019                15957
2024-02-16-10.30           1946506548                88072                22101
2024-02-16-10.35           2002241583                 2188               915101
2024-02-16-10.40           2041460736                 2228               916275
2024-02-16-10.45           1598583529                 2653               602556
2024-02-16-10.50            962043341                 2574               373754
2024-02-16-11.10            200146986               103994                 1924
2024-02-16-11.15            605344554               157991                 3831
2024-02-16-11.20           1897099233                 2222               853780
2024-02-16-11.25           2172573621                 2279               953301
2024-02-16-12.20            125364603                43877                 2857
2024-02-16-12.25              1882371                75559                   24
2024-02-16-12.30           1251490024               142673                 8771
2024-02-16-12.35           1932938072                 2269               851889
2024-02-16-12.40           1792345100                 5088               352269
2024-02-16-12.50            144696810                50607                 2859
2024-02-16-13.00            119962115               134395                  892
```

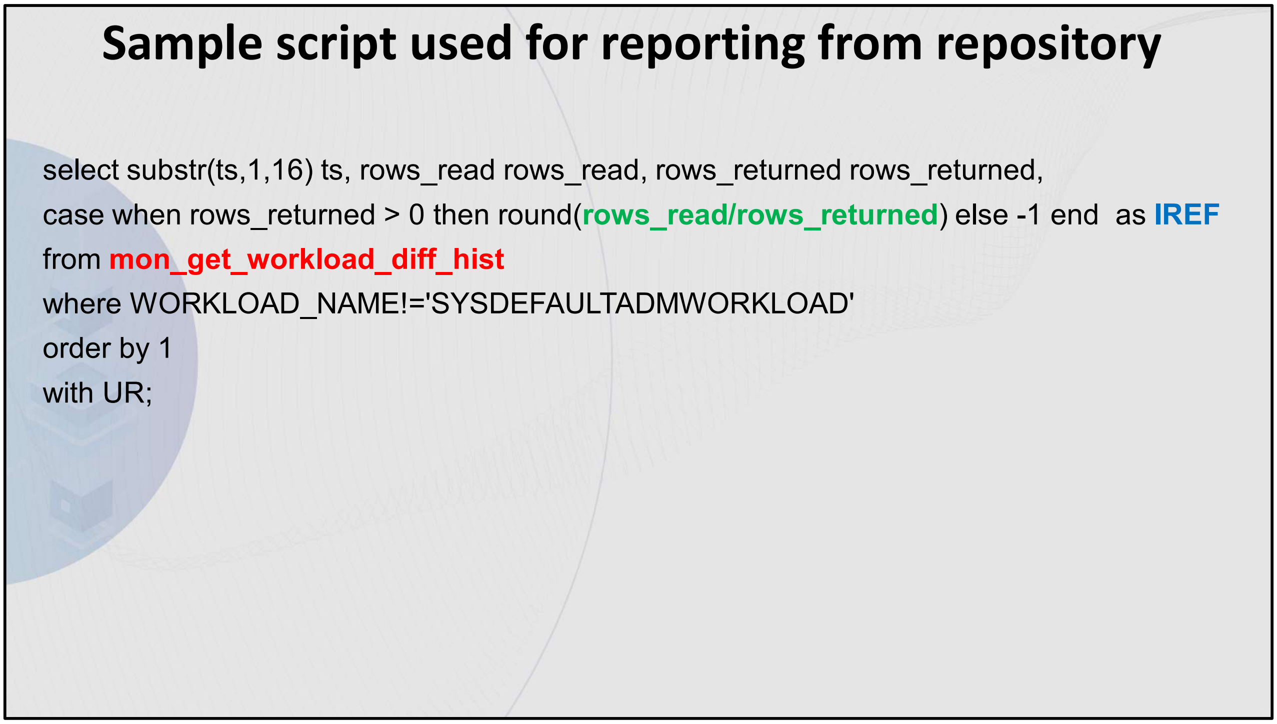
- Plot



IREF is the single most important PKI in Db2 (developed by Db2 community respectful Scott Hayes)

The formula is: IREF = Rows read / Rows Selected (Fetched)

It can be at statement level or database level

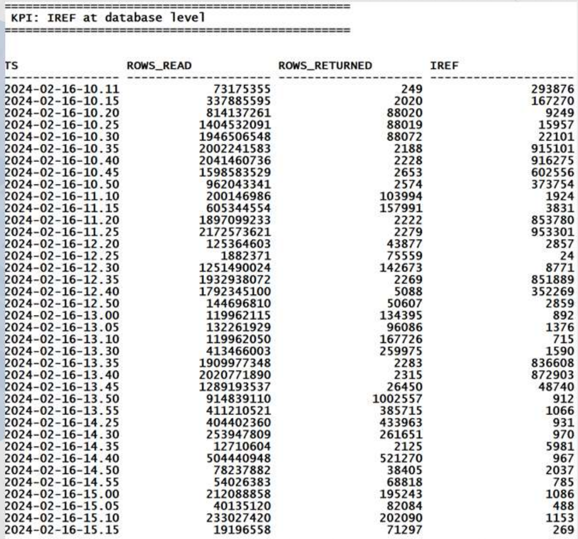


Sample script used for reporting from repository

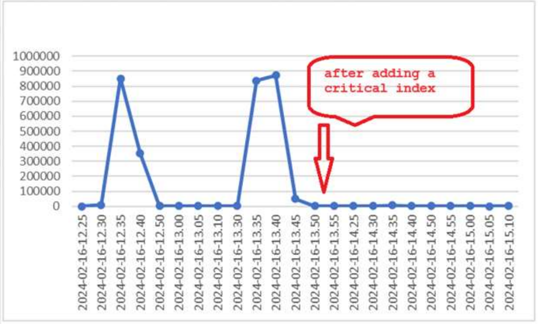
```
select substr(ts,1,16) ts, rows_read rows_read, rows_returned rows_returned,
case when rows_returned > 0 then round(rows_read/rows_returned) else -1 end  as IREF
from mon_get_workload_diff_hist
where WORKLOAD_NAME!='SYSDEFAULTADMWORKLOAD'
order by 1
with UR;
```

Query against history table for a give time frame

Use case #2: Measure the change



```
==============================================
 KPI: IREF at database level
==============================================


TS               ROWS_READ            ROWS_RETURNED        IREF
---------------- -------------------- -------------------- --------------------
2024-02-16-10.11             73175355                  249               293876
2024-02-16-10.15            337885595                 2020               167270
2024-02-16-10.20            814137261                88020                 9249
2024-02-16-10.25           1404532091                88019                15957
2024-02-16-10.30           1946506548                88072                22101
2024-02-16-10.35           2002241583                 2188               915101
2024-02-16-10.40           2041460736                 2228               916275
2024-02-16-10.45           1598583529                 2653               602556
2024-02-16-10.50            962043341                 2574               373754
2024-02-16-11.10            200146986               103994                 1924
2024-02-16-11.15            605344554               157991                 3831
2024-02-16-11.20           1897099233                 2222               853780
2024-02-16-11.25           2172573621                 2279               953301
2024-02-16-12.20            125364603                43877                 2857
2024-02-16-12.25              1882371                75559                   24
2024-02-16-12.30           1251490024               142673                 8771
2024-02-16-12.35           1932938072                 2269               851889
2024-02-16-12.40           1792345100                 5088               352269
2024-02-16-12.50            144696810                50607                 2859
2024-02-16-13.00            119962115               134395                  892
2024-02-16-13.05            132261929                96086                 1376
2024-02-16-13.10            119962050               167726                  715
2024-02-16-13.30            413466003               259975                 1590
2024-02-16-13.35           1909977348                 2283               836608
2024-02-16-13.40           2020771890                 2315               872903
2024-02-16-13.45           1289193537                26450                48740
2024-02-16-13.50            914839110              1002557                  912
2024-02-16-13.55            411210521               385715                 1066
2024-02-16-14.25            404402360               433963                  931
2024-02-16-14.30            253947809               261651                  970
2024-02-16-14.35             12710604                 2125                 5981
2024-02-16-14.40            504440948               521270                  967
2024-02-16-14.50             78237882                38405                 2037
2024-02-16-14.55             54026383                68818                  785
2024-02-16-15.00            212088858               195243                 1086
2024-02-16-15.05             40135120                82084                  488
2024-02-16-15.10            233027420               202090                 1153
2024-02-16-15.15             19196558                71297                  269
```

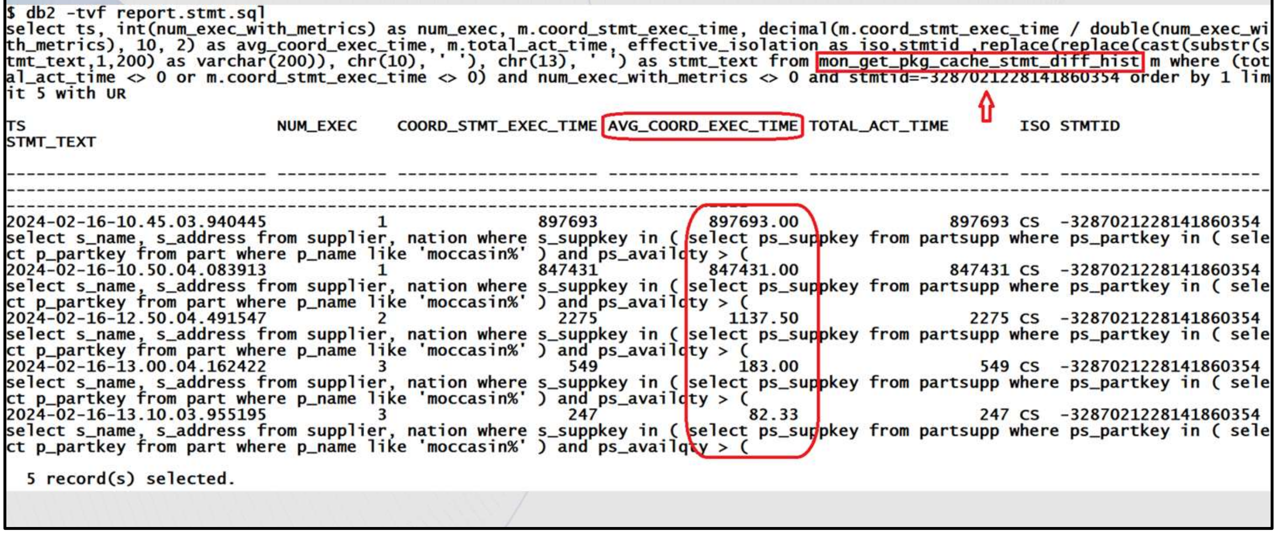


after adding a critical index

Same system used in #1, after adding an index

Use case #3: Identify potential tuning area

- In this example, a long running query identified and tuned

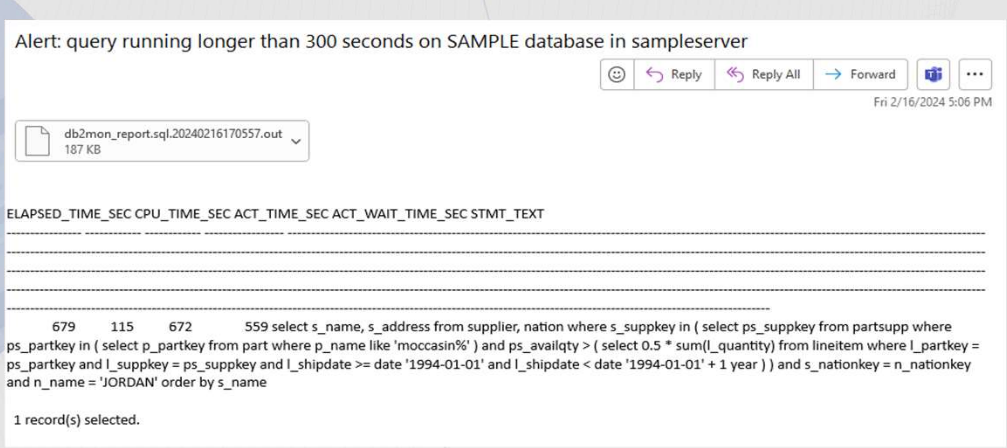


```
$ db2 -tvf report.stmt.sql
select ts, int(num_exec_with_metrics) as num_exec, m.coord_stmt_exec_time, decimal(m.coord_stmt_exec_time / double(num_exec_wi
th_metrics), 10, 2) as avg_coord_exec_time, m.total_act_time, effective_isolation as iso,stmtid ,replace(replace(cast(substr(s
tmt_text,1,200) as varchar(200)), chr(10), ' '), chr(13), ' ') as stmt_text from mon_get_pkg_cache_stmt_diff_hist m where (tot
al_act_time <> 0 or m.coord_stmt_exec_time <> 0) and num_exec_with_metrics <> 0 and stmtid=-3287021228141860354 order by 1 lim
it 5 with UR

TS                         NUM_EXEC    COORD_STMT_EXEC_TIME AVG_COORD_EXEC_TIME TOTAL_ACT_TIME       ISO STMTID
STMT_TEXT

-------------------------- ----------- -------------------- ------------------- -------------------- --- --------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
2024-02-16-10.45.03.940445           1               897693           897693.00               897693 CS  -3287021228141860354
select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where ps_partkey in ( sele
ct p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > (
2024-02-16-10.50.04.083913           1               847431           847431.00               847431 CS  -3287021228141860354
select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where ps_partkey in ( sele
ct p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > (
2024-02-16-12.50.04.491547           2                 2275             1137.50                 2275 CS  -3287021228141860354
select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where ps_partkey in ( sele
ct p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > (
2024-02-16-13.00.04.162422           3                  549              183.00                  549 CS  -3287021228141860354
select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where ps_partkey in ( sele
ct p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > (
2024-02-16-13.10.03.955195           3                  247               82.33                  247 CS  -3287021228141860354
select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where ps_partkey in ( sele
ct p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > (

  5 record(s) selected.
```

Showing an expensive query being identified and tuned from history report.

Use case #4 (alert)

- Alert based on threshold



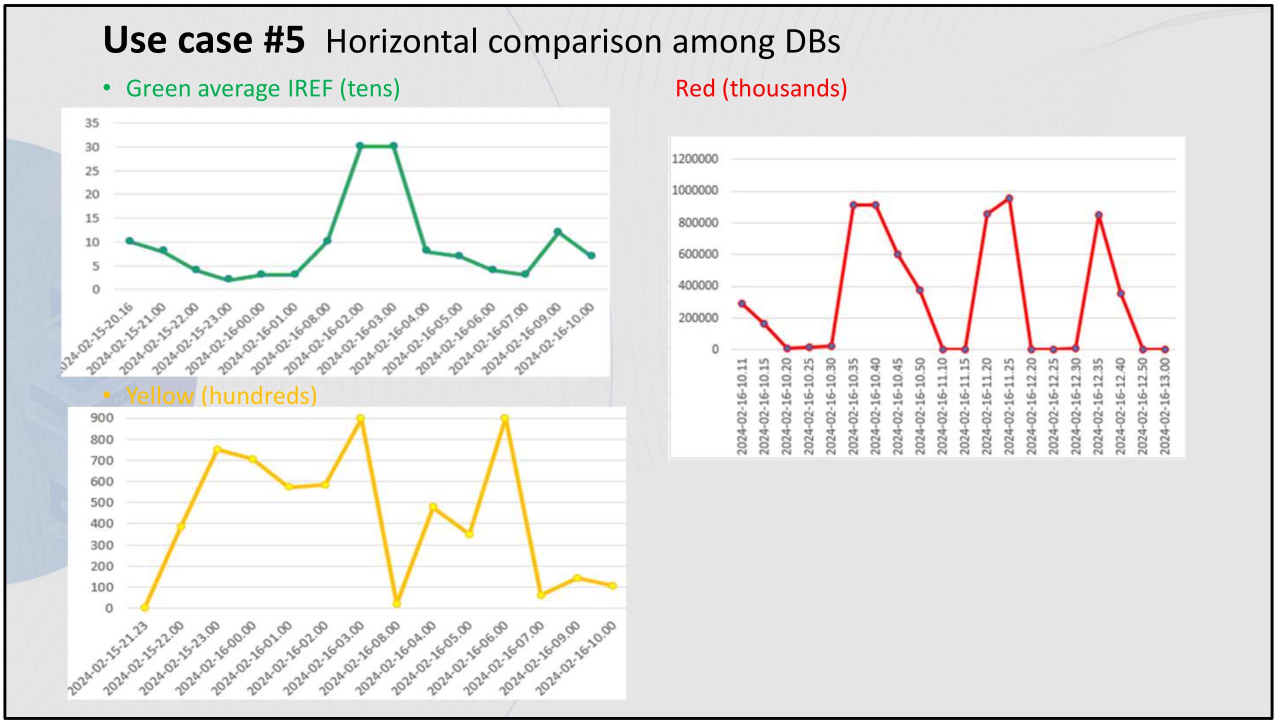
Alert: query running longer than 300 seconds on SAMPLE database in sampleserver

Fri 2/16/2024 5:06 PM

db2mon_report.sql.20240216170557.out
187 KB

```
ELAPSED_TIME_SEC CPU_TIME_SEC ACT_TIME_SEC ACT_WAIT_TIME_SEC STMT_TEXT
---------------- ------------ ------------ ----------------- ------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------
             679          115          672               559 select s_name, s_address from supplier, nation where s_suppkey in ( select ps_suppkey from partsupp where
ps_partkey in ( select p_partkey from part where p_name like 'moccasin%' ) and ps_availqty > ( select 0.5 * sum(l_quantity) from lineitem where l_partkey =
ps_partkey and l_suppkey = ps_suppkey and l_shipdate >= date '1994-01-01' and l_shipdate < date '1994-01-01' + 1 year ) ) and s_nationkey = n_nationkey
and n_name = 'JORDAN' order by s_name

  1 record(s) selected.
```

For example, if query running longer than 300 (either long running or lock wait)



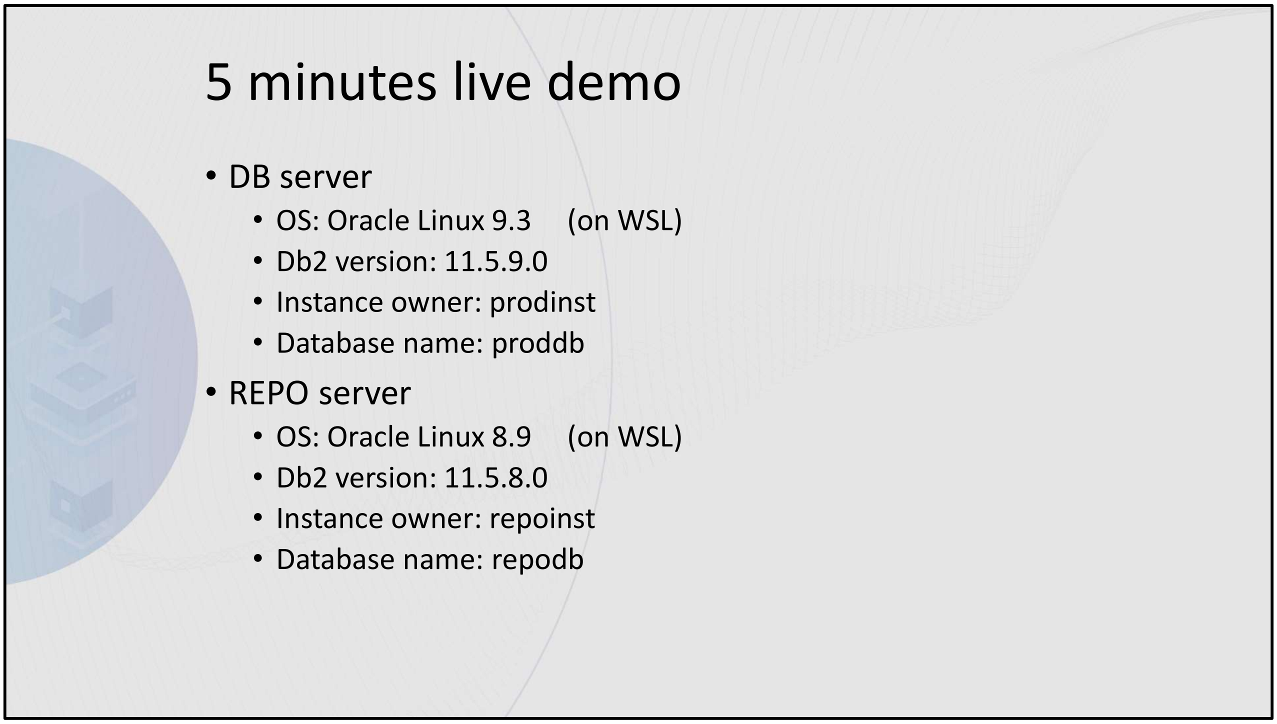
Use case #5 Horizontal comparison among DBs

- Green average IREF (tens)

Red (thousands)

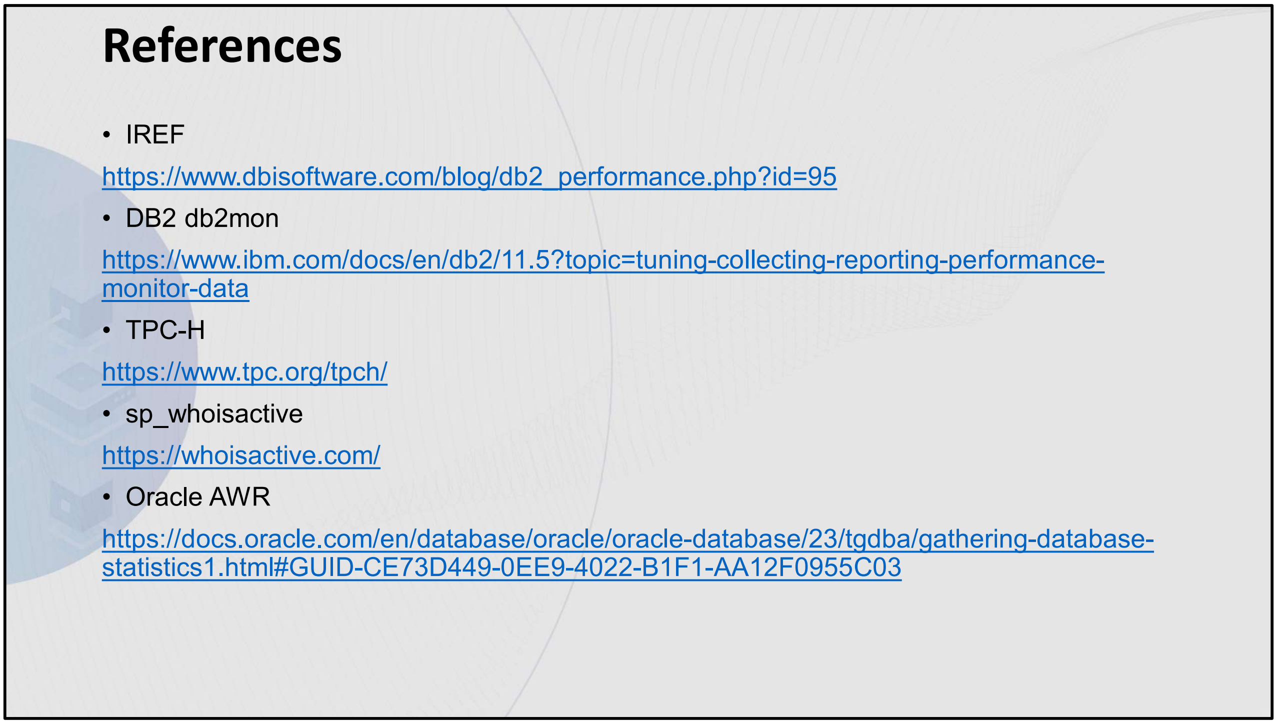
- Yellow (hundreds)

Cross comparison between databases/systems



5 minutes live demo

- DB server
  - OS: Oracle Linux 9.3   (on WSL)
  - Db2 version: 11.5.9.0
  - Instance owner: prodinst
  - Database name: proddb
- REPO server
  - OS: Oracle Linux 8.9   (on WSL)
  - Db2 version: 11.5.8.0
  - Instance owner: repoinst
  - Database name: repodb



References

- IREF

https://www.dbisoftware.com/blog/db2_performance.php?id=95

- DB2 db2mon

https://www.ibm.com/docs/en/db2/11.5?topic=tuning-collecting-reporting-performance-monitor-data

- TPC-H

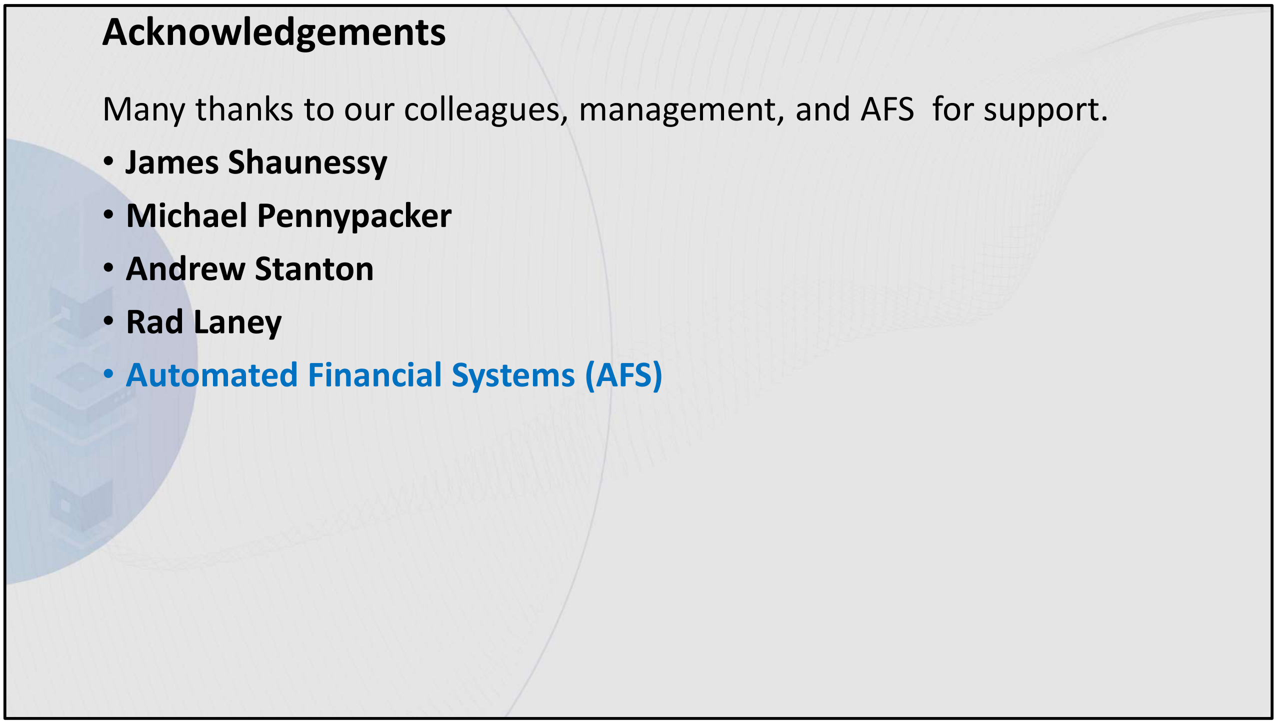
https://www.tpc.org/tpch/

- sp_whoisactive

https://whoisactive.com/

- Oracle AWR

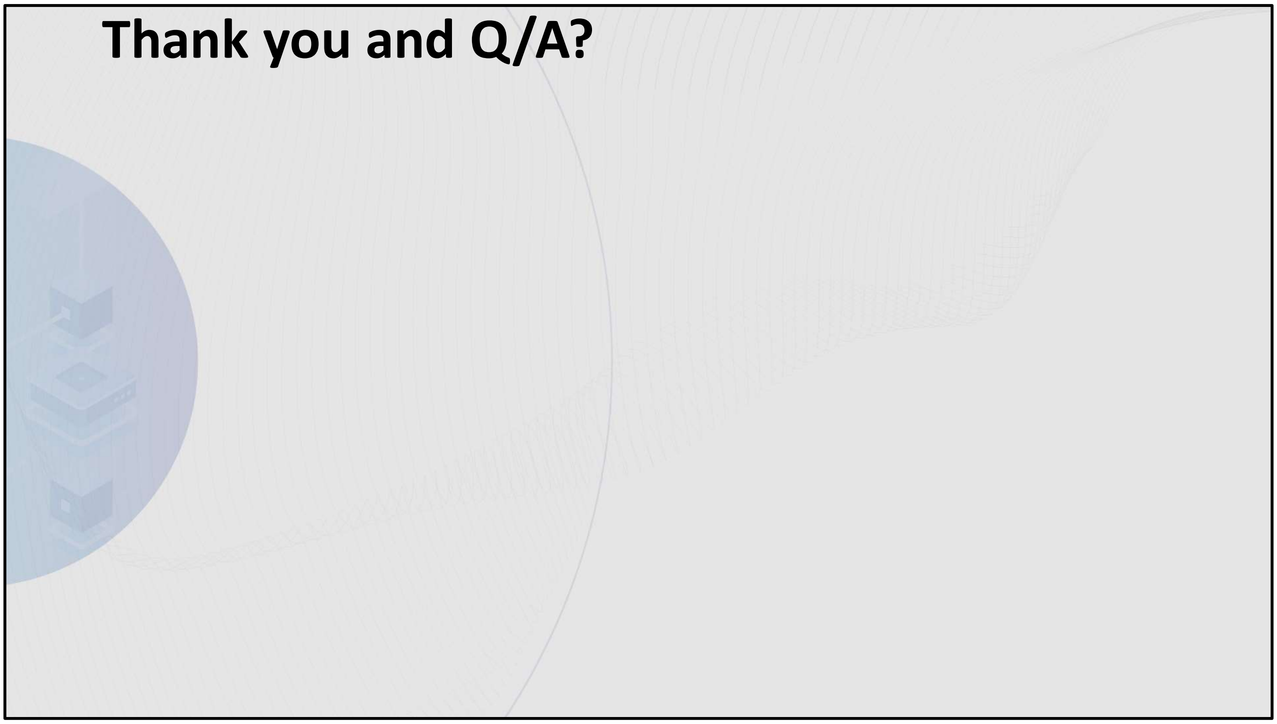
https://docs.oracle.com/en/database/oracle/oracle-database/23/tgdba/gathering-database-statistics1.html#GUID-CE73D449-0EE9-4022-B1F1-AA12F0955C03



Acknowledgements

Many thanks to our colleagues, management, and AFS for support.

- **James Shaunessy**
- **Michael Pennypacker**
- **Andrew Stanton**
- **Rad Laney**
- **Automated Financial Systems (AFS)**



Thank you and Q/A?

What enhancements you want from IBM?
Bugs.

IDUG
2024 NA Db2 Tech Conference
IDUG
@IDUGDb2
#IDUG_NA24

**Building a Db2 "AWR"
(Automatic Workload Repository)
Using Db2 scripts**

Wayne Zhu

wzhu@afsvision.com

AUTO2

Please fill out your session evaluation!

About Wayne Zhu
Sr. Application Engineer, Database SME at AFS
Specialized in Db2, Oracle, SQL Server, MySQL, PostgreSQL, and MongoDB
Former IBM Champion

About Kirk Spadt
Director of Architecture at AFS

About Automated Financial Systems
Automated Financial Systems, Inc. (AFS) is the industry leader in commercial lending and credit solutions for financial institutions