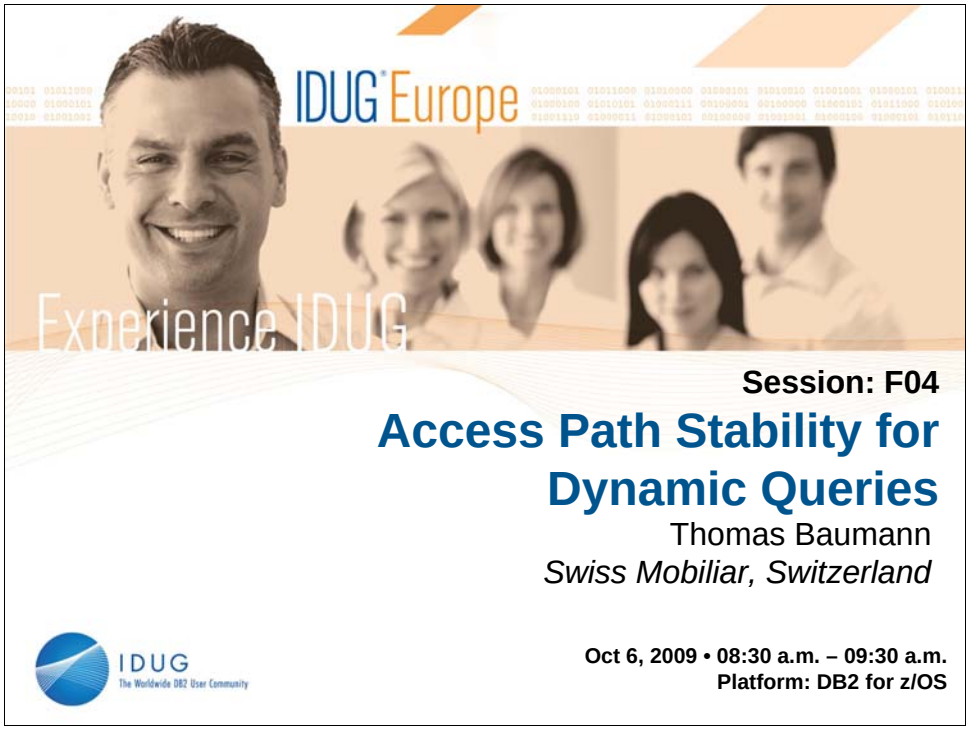
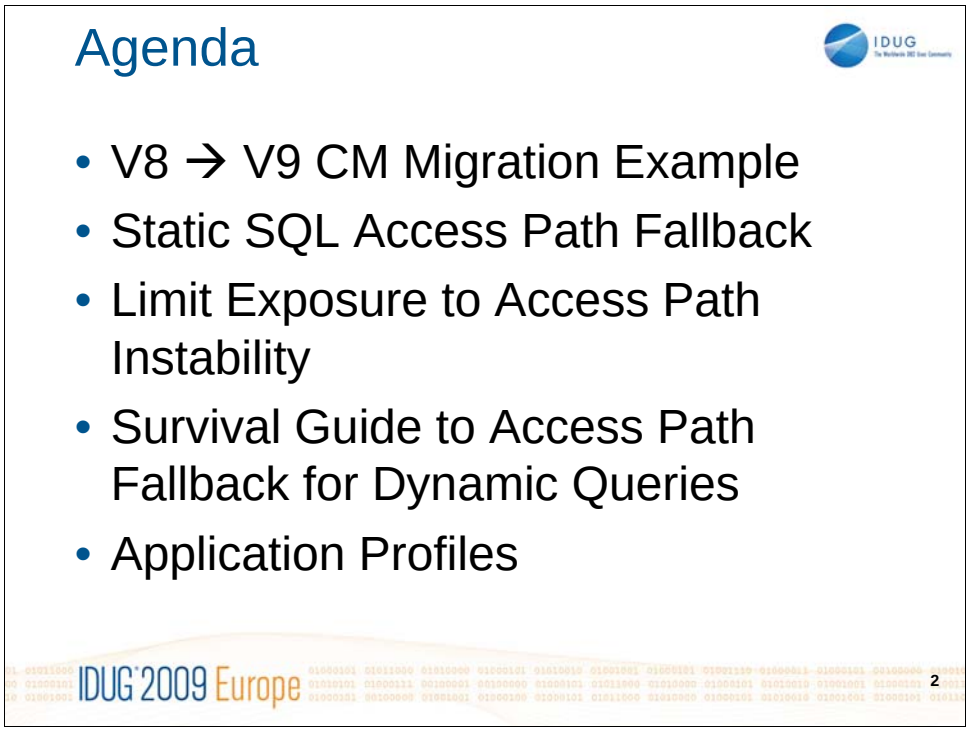


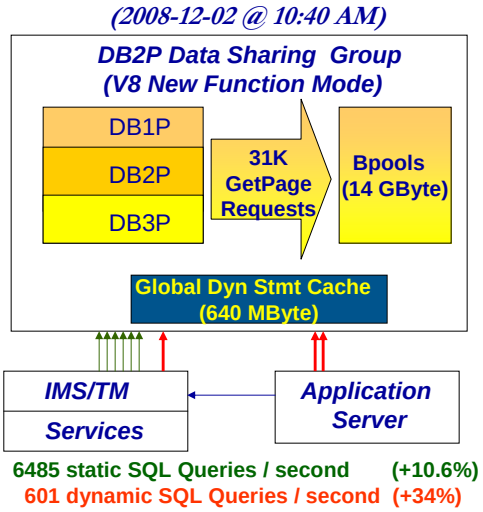
To Rebind or not after a new DB2 release, maintenance upgrade, or after running RUNSTATS - this has been a crucial question for a long time. After the implementation of optimizer hints, access plan management, and the REOPT bind option, things became easier for static SQL. But concerning dynamic SQL, it is still at the customer's responsibility to ensure access path stability. This presentation shows methods for your dynamic workload to avoid access path regression and to correct unexplainable access paths, such that you don't dread your next RUNSTATS run, application release or maintenance upgrade



The following key points will be discussed:

- Causes of access path instability
- Access Path Fallback for static queries
- Limiting exposure to instability for dynamic queries
- Access Path corrections for dynamic queries
- Query Performance Management as a continuous process

Swiss Mobiliar: Key Facts at a Glance



- Major insurer for all sectors.
- Leading property insurer.
- Leader in term insurance (private and occupential pensions).
- Reliable and expert partner.
- Mutual basis.
- Customers and employees share in our success.
- Continuity and a lasting relationship with policyholders.
- Committed to good corporate citizenship for the common good and the environment.

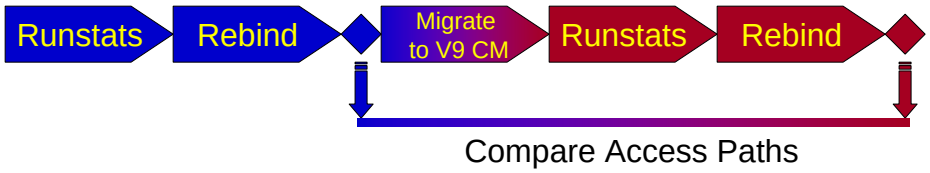
Disclaimer

- The Information contained in this presentation has not been submitted to any formal Swiss Mobiliar or other review and is distributed on an ‘as is‘ basis without any warranty either expressed or implied. The use of this information is the user‘s responsibility.
- The procedures, results and measurements presented in this paper were run in either the test and development environment or in the production environment at Swiss Mobiliar in Berne, Switzerland. There is no guarantee that the same or similar results will be obtained elsewhere. Users attempting to adapt these procedures and data to their own environments do so at their own risk. All procedures presented have been designed and developed for educational purposes only.

A standard disclaimer.

V8 → V9 CM Migration

- How much Change for Static Queries?
  - Statistics on Access Path Changes

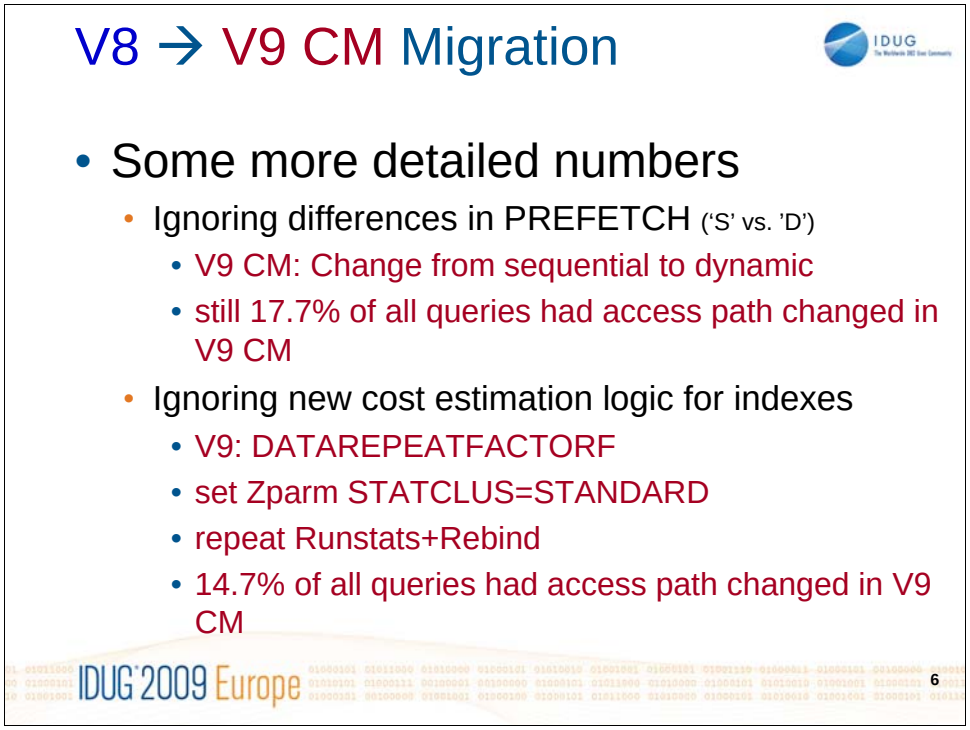


  - 22793 Queries compared
  - 4855 Queries with changed access paths (21.3%)
  - How to get these numbers see notes page


```
CREATE TABLE AQT230.APTABLE (
   LINENO INTEGER GENERATED ALWAYS as identity
  ,PROGNAME CHAR(28)
  ,VERSION CHAR(28)
  ,COLLID  CHAR(28)
  ,QUERYNO INTEGER
  ,METHOD  smallint
  ,CREATOR VARCHAR(128) NOT NULL
  ,TNAME VARCHAR(128) NOT NULL
  ,ACCESSTYPE CHAR(2) NOT NULL
  ,MATCHCOLS smallint NOT NULL
  ,ACCESSCREATOR VARCHAR(128) NOT NULL
  ,ACCESSNAME VARCHAR(128) NOT NULL
  ,INDEXONLY CHAR(1) NOT NULL
  ,PREFETCH CHAR(1) NOT NULL WITH DEFAULT);
create index AQT230.aptable1 on AQT230.aptable (accesscreator,
accessname) ;
create index AQT230.xaptable2 on AQT230.aptable(progname, collid,
version, queryno);
CREATE TABLE AQT230.APID (
        PROGNAME CHAR(28)
       ,VERSION CHAR(28)
       ,COLLID  CHAR(28)
       ,QUERYNO INTEGER
       ,APID    LONG VARCHAR );
create index AQT230.xapid on AQT230.apid(progname, collid, version,
queryno);
```

LOAD PLAN_TABLE data into APTABLE, then INSERT concatenation of APTABLE rows per query into APID

For further details of these steps, please e-mail to thomas.baumann@mobi.ch



In DB2 z/OS V9.1, sequential prefetch will be replaced by dynamic prefetch in many cases. If we interpret an access path which differs in the prefetch column only (change from 'S' to 'D') as identical, the number of changed access paths decreases from 21.3% to 17.7%.

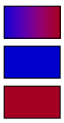
For indexes, that contain either many duplicate key values or key values that are highly clustered in reverse order, cost estimation that is based purely on CLUSTERRATIOF can lead to repetitive index scans. In the worst case, an entire page could be scanned one time for each row in the page. DB2 access path selection can avoid this performance problem by using a cost estimation formula based on DATAREPEATFACTORF statistic to choose indexes. Whether DB2 will use this statistic depends on the value of the STATCLUS subsystem parameter:

STATCLUS=STANDARD: Runstats sets DATAREPEATFACTORF=-1
STATCLUS=ENHANCED: Runstats updates DATAREPEATFACTORF

STATCLUS=ENHANCED is the default value for V9.1

V8 → V9 CM Migration

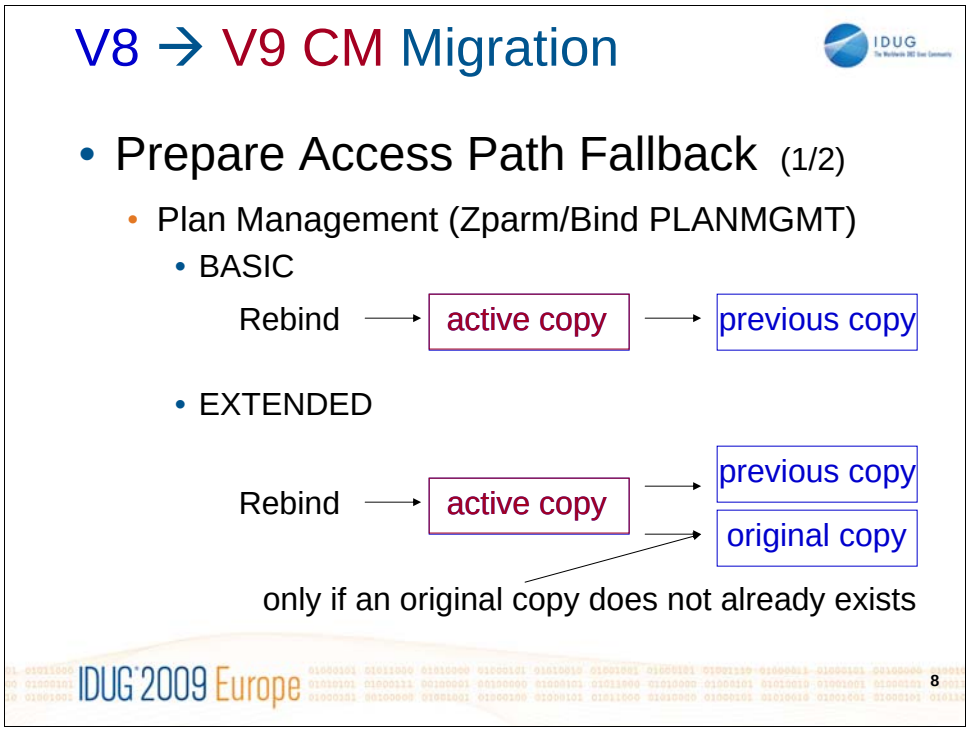
- How many different access paths for the 22793 queries exist?
  - V8: 4535 access paths
  - V9: 4546 access paths



3581 access paths used in V8 and V9
954 access paths not longer used in V9
965 new access paths in V9 CM

  - See notes page for detail queries
- There is a need for a fallback option

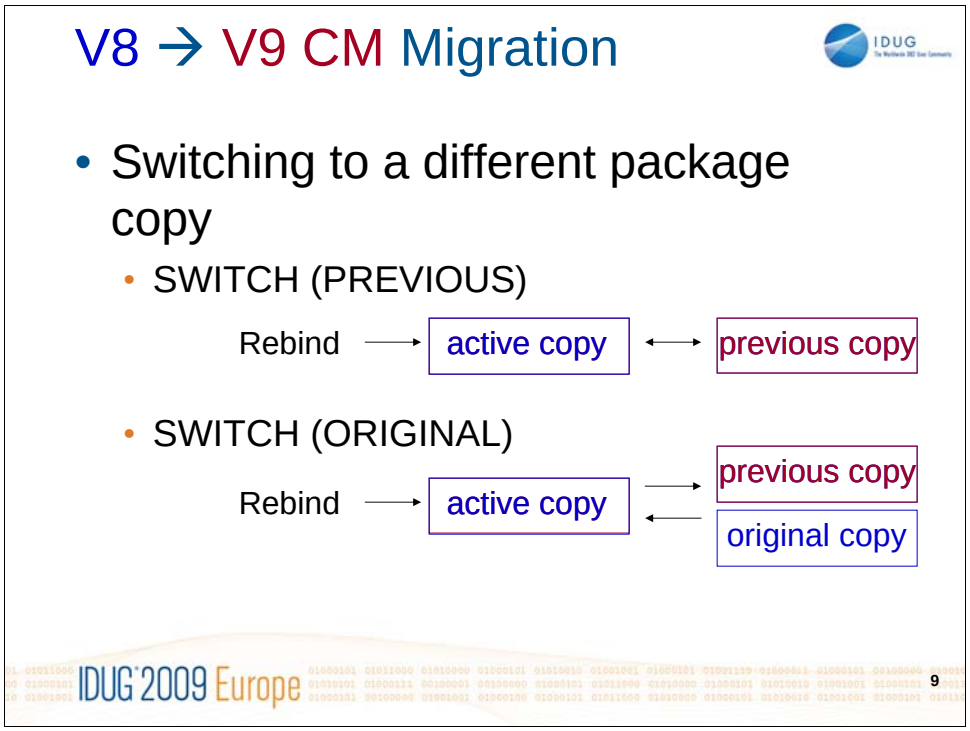

```
-- SELECT NO OF QUERIES WITH CHANGED ACCESS PATHS
SELECT COUNT(*)
FROM AQT230.APID_V8 V8, AQT230.APID_V9 V9
WHERE  V8.PROGNAME= V9.PROGNAME
  AND  V8.COLLID  = V9.COLLID
  AND  V8.VERSION = V9.VERSION
  AND  V8.QUERYNO = V9.QUERYNO
  AND  V8.APID   <> V9.APID


-- SHOW OLD AND NEW ACCESS PATHS FOR QUERIES
-- WITH CHANGED ACCESS PATHS
SELECT V8.PROGNAME, V8.COLLID, V8.VERSION,
       V8.QUERYNO, V8.APID AS AP_OLD,
       V9.APID AS AP_NEW
FROM – same as query above


-- SELECT NO OF DIFFERENT ACCESS PATHS
SELECT COUNT(DISTINCT APID) FROM AQT230.APID_V8;
SELECT COUNT(DISTINCT APID) FROM AQT230.APID_V9;
```



You can use package copies to automatically save pertinent catalog table and directory records when you rebind an existing package. If a performance regression occurs after you rebind a package, you can fall back to the older copy of the package.

You can use package copies with regular packages, non-native SQL procedures, external procedures and trigger packages.

Issue a REBIND PACKAGE or REBIND TRIGGER PACKAGE command, and specify the PLANMGMT option.

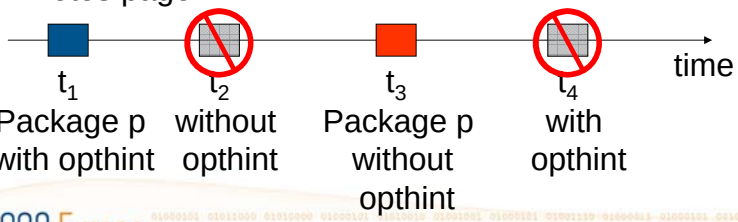


If you use package copies when you rebind a package, you can revert to a saved copy of the package if you encounter performance regression after rebinding the package.

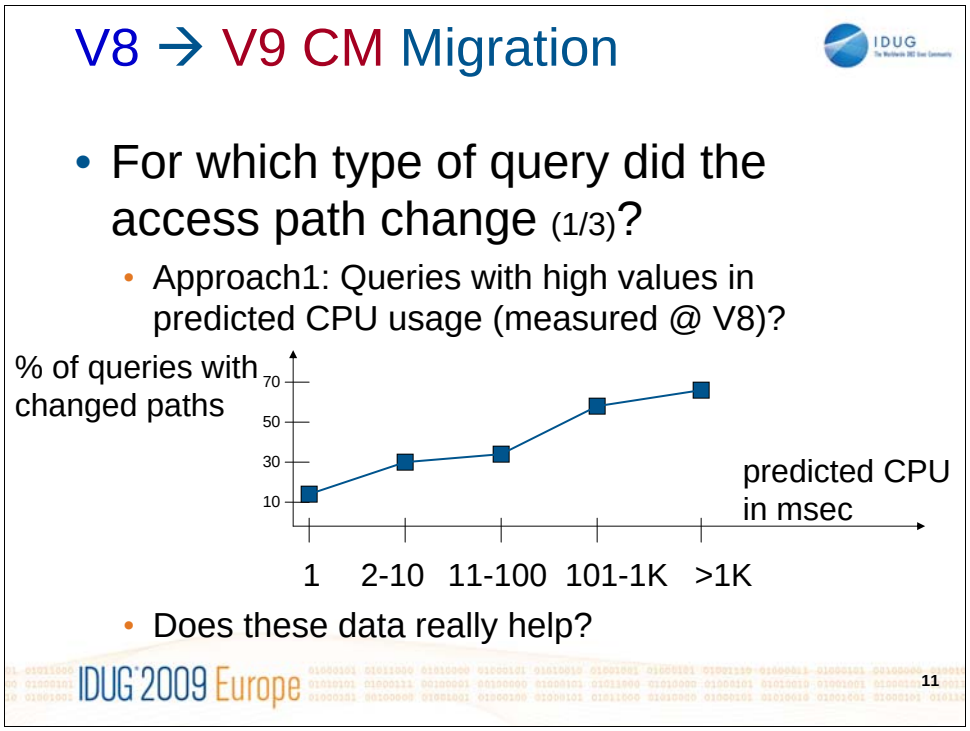
You can switch to a previous copy of the package only if you specified PLANMGMT (BASIC) or (EXTENDED) with the REBIND PACKAGE or REBIND TRIGGER PACKAGE statement. You can switch to an original copy of the package only if you specified PLANMGMT (EXTENDED).

V8 → V9 CM Migration

- Prepare Access Path Fallback (2/2)
  - Optimizer Hints
    - might be necessary after new BIND
    - be aware of new queryno
    - See Check-Query to detect “lost” optimizer hints after new bind or rebind at t_3 of same package in notes page:




```
SELECT DISTINCT P1.NAME, P3.BINDTIME, P3.CREATOR
FROM SYSIBM.SYSPACKAGE P1, SYSIBM.SYSPACKAGE P3
WHERE P1.NAME=P3.NAME
  AND P1.OPTHINT<>''
  AND P3.OPTHINT= ''
  AND P1.BINDTIME < P3.BINDTIME
  AND P3.BINDTIME > '2009-01-07-00.00.00.000000'
  AND NOT EXISTS
  (SELECT 1 FROM SYSIBM.SYSPACKAGE P2
  WHERE P1.NAME=P2.NAME
    AND P2.OPTHINT=''
    AND P1.BINDTIME < P2.BINDTIME
    AND P1.COLLID   = P2.COLLID
    AND P3.COLLID   = P2.COLLID
    AND DATE(P2.BINDTIME) < DATE(P3.BINDTIME))
  AND NOT EXISTS
  (SELECT 1 FROM SYSIBM.SYSPACKAGE P4
  WHERE P3.NAME=P4.NAME
    AND P4.OPTHINT <> ''
    AND P3.BINDTIME < P4.BINDTIME
    AND P3.COLLID   = P4.COLLID
    AND DATE(P3.BINDTIME) < DATE(P4.BINDTIME))   ;
```



The diagram shows that queries with a high predicted CPU usage were more likely to change access paths at V9 migration. This is good news (these access paths should become faster after migration), but there is still a high percentage of changes (13%) for queries with PROCMS=1 as predicted CPU time.

The following query reports the number of changed access paths for one of the CPU-prediction-intervals shown in the diagram:

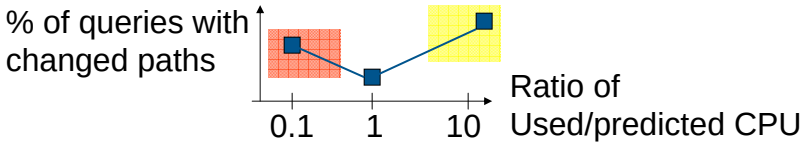
```
SELECT COUNT(*)
FROM AQT230.APID_V8 V8, AQT230.APID_V9 V9,
DB2WVIEW.DSN_STATEMNT_TABLE PROC1
WHERE  V8.PROGNAME=V9.PROGNAME
  AND  V8.COLLID  =V9.COLLID
  AND  V8.VERSION =V9.VERSION
  AND  V8.QUERYNO =V9.QUERYNO
  AND  V8.PROGNAME=PROC1.PROGNAME
  AND  V8.QUERYNO =PROC1.QUERYNO
  AND  V8.COLLID  =PROC1.COLLID
  AND DATE(PROC1.EXPLAIN_TIME) = '20.11.2008'
  AND PROC1.PROCMS BETWEEN 2 AND 10
  AND  V8.APID   <> V9.APID
```

V8 → V9 CM Migration

- For which type of query did the access path change (2/3)?
  - Approach2: Queries with a small difference in predicted CPU usage between the *original prediction*, and a prediction *if the first index chosen would not be available*?
  - Might be OK, but hard to compute!
    - Need a much smaller set of candidate queries

V8 → V9 CM Migration

- For which type of query did the access path change (3/3)?
  - Approach3: Queries with differences between *predicted CPU* and *used CPU at runtime*
    - Uses much more CPU than predicted
    - Uses much less CPU than predicted



Queries with a large difference in predicted vs. used CPU, either with a

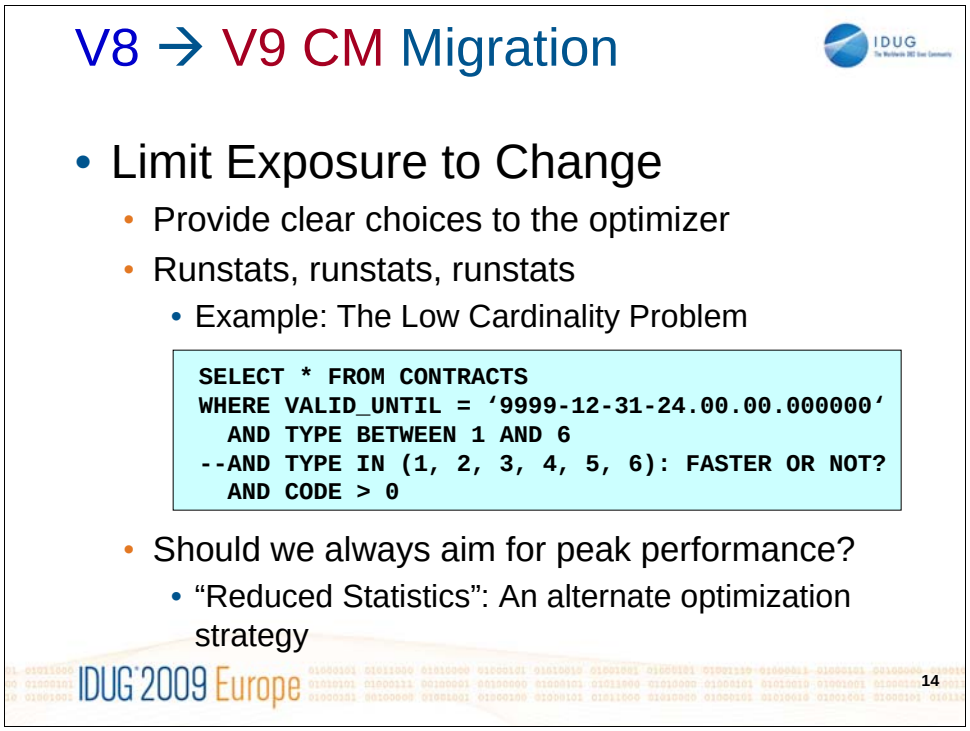
- high ratio (> 5) of used/predicted CPU, or
- low ratio (< 0.5) of used/predicted CPU

are most likely to have the access path changed. Queries that run slower than predicted are candidates for query tuning analysis, independent of maintenance upgrade, release change or updated catalog statistics.

From the perspective of access path stability, queries which run faster than predicted need more attention, because an access path change might increase the response time of such queries.

There are many possible reasons for large differences in prediction vs. Run time, among others:

- unknown data skew
- lack of correlation statistics
- optimistic range predicate estimates
- lack of knowledge of number of required rows

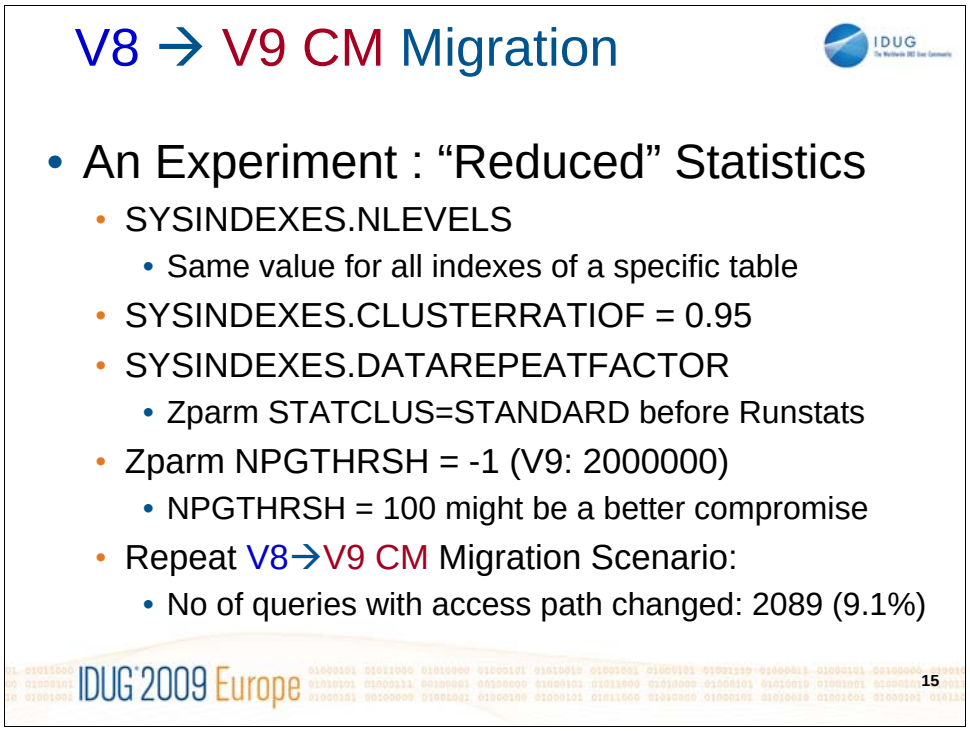


The query shown above could benefit from V9.1's new histogram statistics, and therefore provide a better access path once these statistics data has been collected:

RUNSTATS .... HISTOGRAM [NUMQUANTILES 10]

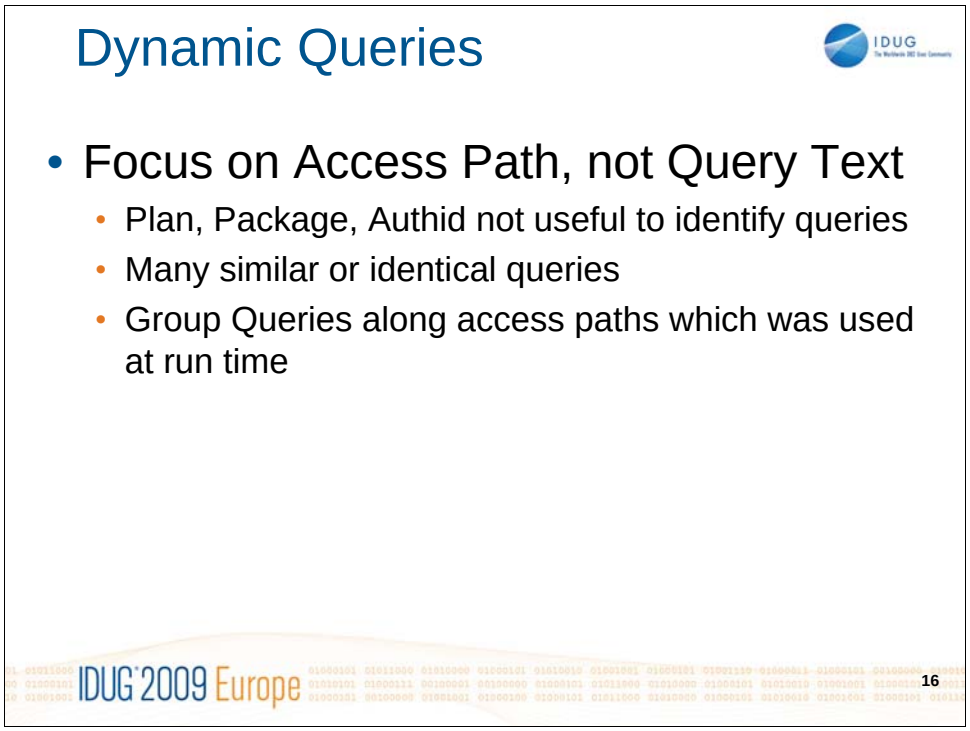
As there are only a few rows in the contracts table with values of TYPE outside the selected scope (but with values such as 2000), histogram statistics would clearly show that a tablespace scan would be much more efficient than an access path using an index on the TYPE column.

Until V8, it was more efficient to code explicit values in an IN list in order to direct the optimizer to statistics on discrete values.



The following query updates SYSIBM.SYSINDEXES in order to set the same value for all indexes of the same table:

```
create table nlevels (tbcreator char(8),
tbname varchar(128), nlevels integer);
create index xnlevels on nlevels (tbcreator,
tbname);
insert into nlevels (tbcreator, tbname, nlevels)
select tbcreator, tbname, ceiling(avg(nlevels))
from sysibm.sysindexes
where tbcreator IN (...);
group by tbcreator, tbname;
update sysibm.sysindexes  i  set nlevels =
(select n.nlevels from nlevels n where
i.tbcreator=n.tbcreator
 and i.tbname=n.tbname)
where tbcreator IN (...) ;
update sysibm.sysindexes set clusterratiof=0.95
where tbcreator IN (...) ;
```

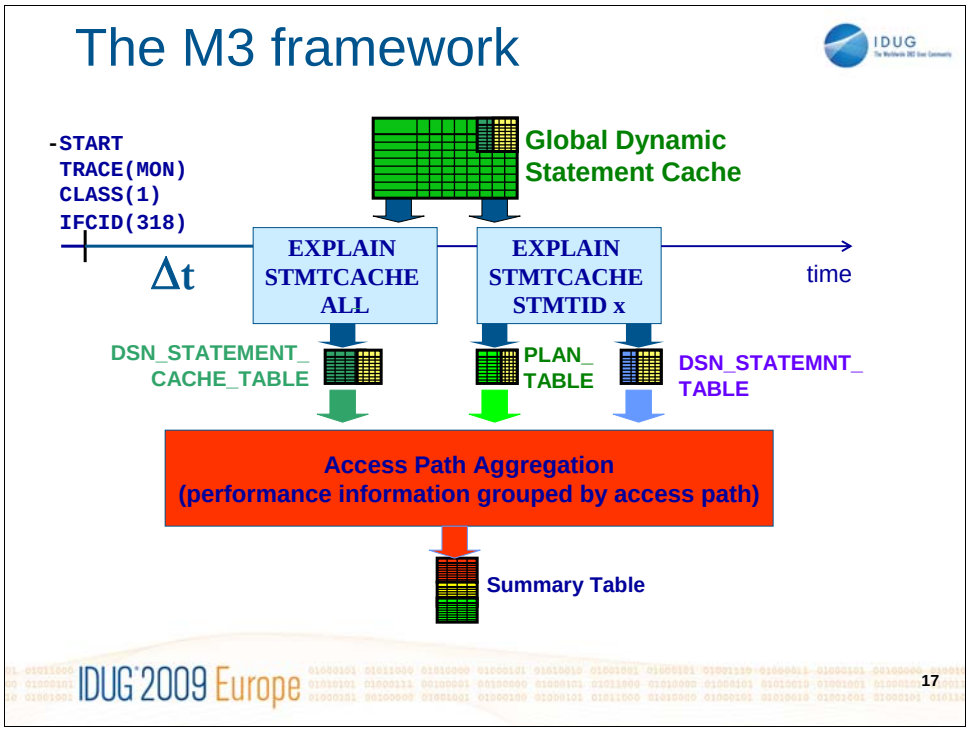


One of the main problems of dynamic SQL tuning is the aggregation of performance information. DB2 for z/OS V8's EXPLAIN STMTCACHE ALL delivers raw data on a very granular level that needs to be aggregated into statement groups.

At Swiss Mobiliar, we group all statements into statement groups according to their access paths used at run time. Thresholds and analysis will be applied to these statement groups: This allows the user to quickly identify the access paths which need attention and improvement, without even looking at the query text itself.

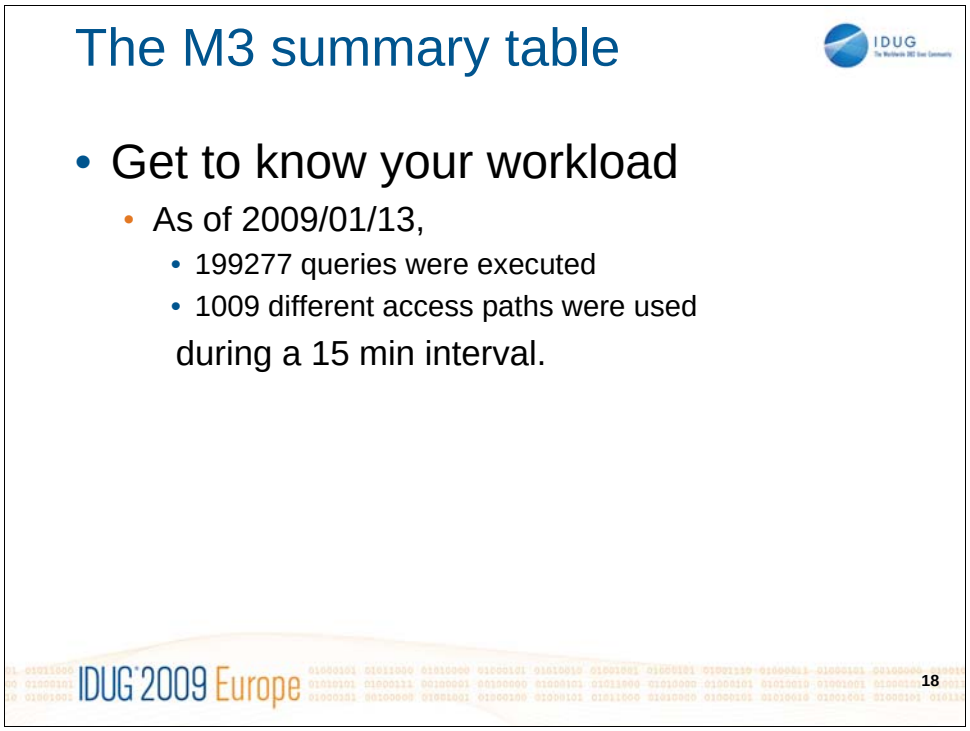
We have developed a framework around the EXPLAIN STMTCACHE ALL query, which aggregates the executed queries among their access paths. We named this framework 'M3' according to a powerful car produced and tuned by BMW. M3 guides the user through a comprehensive tuning methodology for evaluating the performance of a database. Key performance metrics are measured, and the users are given a comprehensive analysis of their dynamic SQL workload and a number of tuning ideas.

M3 is available at no cost at Swiss Mobiliar (e-mail to: thomas.baumann@mobi.ch).



The DSN_STATEMENT_CACHE_TABLE is also populated with a STMTID column, which contains a unique value for each row. Using this value in several EXPLAIN STMTCACHE STMTID queries (one for each stmtid), the PLAN_TABLE and DSN_STATEMNT_TABLE will also be populated – with optimizer information taken directly from the dynamic statement cache. There is no re-evaluation of the access path at this moment, it's just about writing out the stored access path (and cpu consumption preview respectively) information.

Eventually, we join the three tables according to the stmtid, and produce performance information grouped by access path.



If your organization manages dozens of applications and hundreds of databases, wouldn't you like to be able to quickly determine if your dynamic SQL statements require your team's attention?

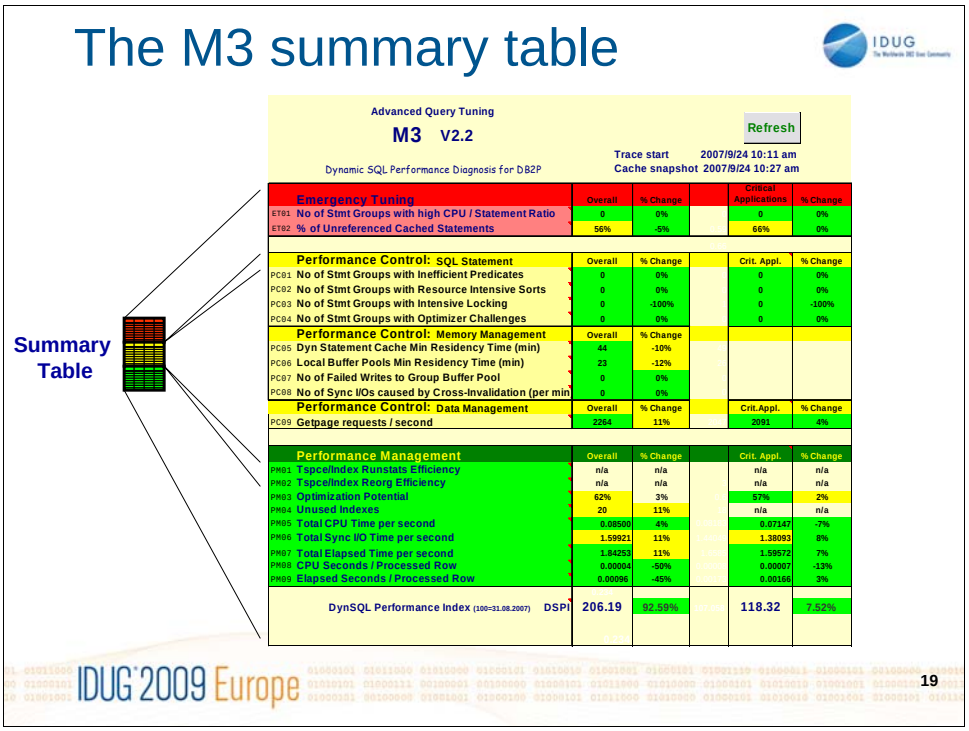
If there is a poor performance or inefficiency problem, M3 will find it and bring it to your attention. M3 guides the user through a comprehensive tuning methodology for evaluating the performance of a database. Key performance metrics are measured, and the users are given a comprehensive analysis of their dynamic SQL workload and a number of tuning ideas.

The overview report represents the latest time interval's measurements, compared with the previous one. It quickly identifies tuning opportunities, and highlights problem regions which require further attention.

A Dynamic SQL Performance Score, like a Credit Score, is also provided so that you can quickly understand your workload's health and efficiency. You no longer need to digest dozens of metrics and indicators.

Detail reports provide further details to the overview report. They can be executed by any SQL query processor and need no further adjustment.

The M3 summary table



The summary table is divided into three different parts:

The first – red – part highlights emergency performance issues which should be addressed immediately, such as queries from hell or repetitive queries with literals instead of parameter markers.

The second – yellow – part identifies access paths which need analysis and probably improvement, but not as an emergency action.

The third - green – part illustrates global performance numbers such as the average cpu usage per query etc.

M3 resources

Sel	Stepname	Procstep	Errortxt	CC	CPU-Time
.	HJRQAS84	QAS84		0	0:00.02
.	M3CLNUP			0	0:01.15
.	M3STATR			0	0:00.02
.	M3COUNT			0	0:00.00
.	M3EXPL0			0	0:03.49
.	M3STOTR			0	0:00.02
.	M3APA1			0	0:00.15
.	M3APA2		HIGH CC	4	0:00.04
.	M3APA3			0	0:00.01
.	M3EXPL1			0	0:16.36
.	M3APTAB	DSNUPROC		0	0:00.41
.	M3APID			0	0:56.26
.	M3FREQTB	DSNUPROC		0	0:00.09
.	M3SUMMRY			0	0:02.50
.	M3CRITIC			0	0:06.78
.	HJRQAS88	QAS88		0	0:00.02

EXPLAIN STMTCACHE ALL (23894 statements)

EXPLAIN STMTCACHE STMTID x (executed 6306 times, once for every statement with stat_exec > 0)

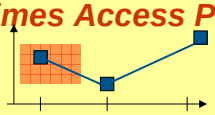
Limit Exposure to Instability

- Analyze Access Paths
  - Access Paths with
    - CPU estimates >> CPU runtime

***estimated CPU** (dsn_statemnt_table)*

measured CPU** (dsn_statement_cache_table)* ***> 2

& No of Times Access Path Used > 1 / sec

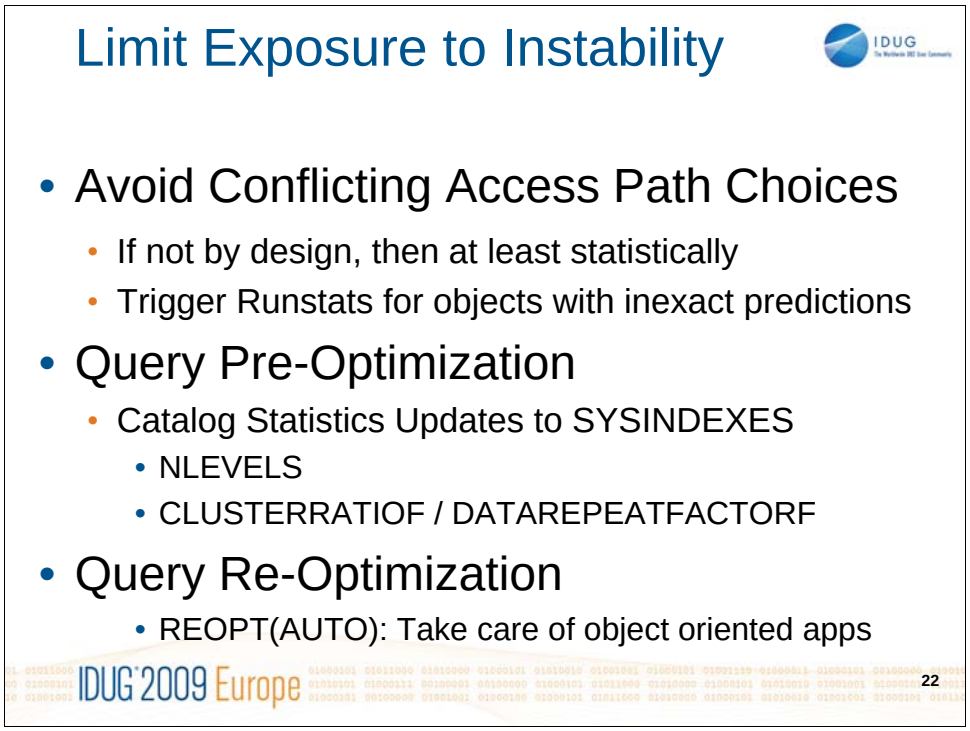


At Swiss Mobiliar as of 2009/01/13, this query reported 44 different access paths (4.3% of all dynamic SQL access paths)

"If you compare the driver's intended direction (by measuring steering angle) to the vehicle's actual direction (by measuring lateral acceleration, vehicle rotation, and individual road wheel speeds) and detect when the vehicle is not going where the driver is steering, DSC (dynamic stability control) intervenes and applies the brakes to individual wheels asymmetrically."
That's what BMW says about electronic stability control.

Translated to DB2 and access stability, it means:

"If you compare the optimizer's estimated access path cost (by measuring dsn_statemnt_table.procsu) to the query's actual performanc numbers (dsn_statement_cache.stat_cpu and other measurements) and detect when the runtime numbers are different from what was predicted, the DBA intervenes and applies corrections to the access path or the statistics collected."

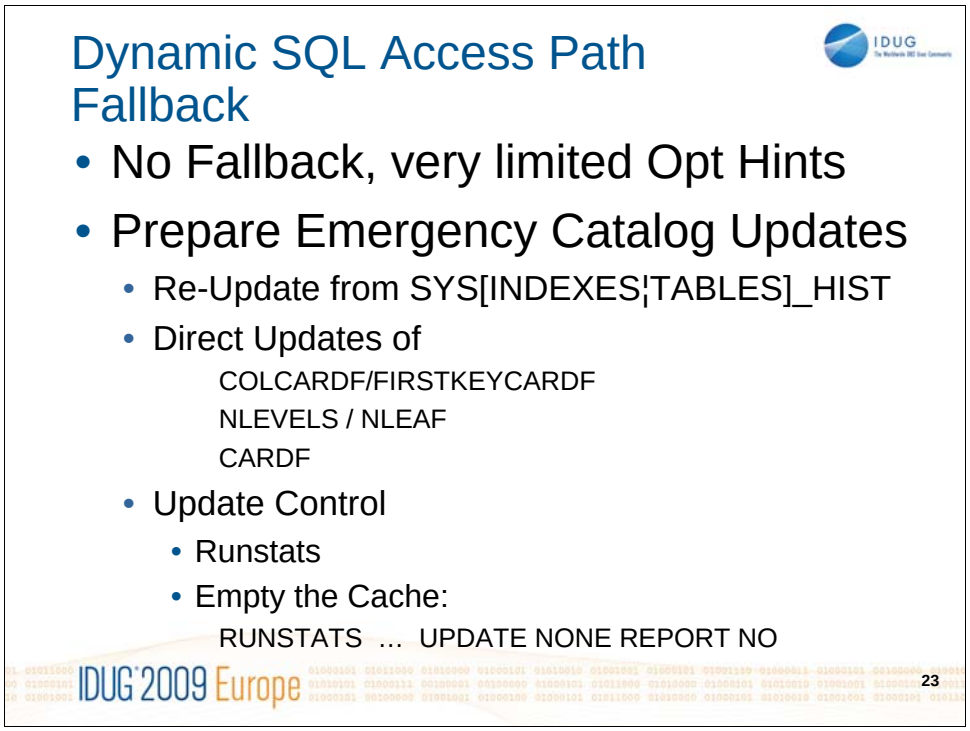


The more the optimizer is challenged, the more likely you are to suffer from access path instability after even minor statistics changes, release or maintenance upgrades.

REOPT(ALWAYS) prepares every execution of a query, which is usually not a useful option on a package level, if there are many statements in a given package. REOPT(ONCE) considers only the first set of literals used for a given query. For object-oriented applications, the first time a query is executed, it uses sometimes specific literals to cache the „common“ values of objects. This first execution of a query is often performed at application startup time only, and the literals used during regular life time are different from that first execution.

REOPT(AUTO) balances the pros and cons of REOPT(ALWAYS) and REOPT(ONCE), and is an interesting option to increase dynamic SQL query performance. It automatically reoptimizes the query when predicate filtering changes dramatically, such that table join sequence or index choice may change.

But like all options which change the behaviour at runtime, this option increases access path instability and should therefore be used carefully.



In contrast to static SQL, where access plan management and optimization hints offer some fallback options, there is no fallback for dynamic queries to the execution mode before the latest change.

So, the best option is to avoid the need for fallback, by avoiding conflicting access patch choices, either by design or statistically.

However, be ready for emergency fallbacks by updating catalog statistics, either by re-updating the values from SYSIBM.SYSINDEXES_HIST and/or SYSTABLES_HIST and/or SYSCOLUMNS_HIST, or by directly updating catalog statistics information in emergency cases.

Dynamic SQL Access Path Fallback

Table Column	default value (w/o Runstats)	used by optimizer to calculate cost for
SYSTABLES		
CARDF	10000	indexscan with low clusterratio
NPAGES	1+ CARDF/20	indexscan with high clusterratio
SYSTABLESPACE		
NACTIVEF	1 + CARDF/20	tablespace scan
SYSCOLUMNS		
COLCARDF	25	FF for ,=, and range predicates
HIGH2KEY	---	FF for range predicates
LOW2KEY	---	FF for range predicates

This survival guide information helps to identify which tables and columns should be updated in order to force a different access path.

Primarily, the updated values should be taken from the corresponding history catalog table (for example, SYSIBM.SYSTABLES_HIST).

Dynamic SQL Access Path Fallback

Table Column	default value (w/o Runstats)	used by optimizer for
SYSINDEXES		
FIRSTKEYCARDF	25	FF for matching index scan
FULLKEYCARDF	25	FF for matching Index Scan
NLEAF	CARDF/300	Estimation index I/O cost
NLEVELS	0	Estimation index I/O cost
CLUSTERRATIOF/ DATAREPEATFACTORF	0	Estimation for tspce I/O cost
SYSCOLDIST(STATS)		
NAME / NUMCOLUMNS	---	skewed data distribution
COLVALUE	---	skewed data distribution
FREQUENCYF	---	skewed data distribution

Dynamic SQL Access Path Fallback

COLCARDF example for range predicates

COLCARDF	Filter-Factor for </>[=]	Filter-Factor for LIKE/BETWEEN
>= 100M	1/10,000	3/100,000
>= 10M	1/3,000	1/10,000
>= 1M	1/1,000	3/10,000
>= 100,000	1/300	1/1,000
>= 10,000	1/100	3/1,000
>= 1,000	1/30	1/100
>= 100	1/10	3/100
>= 2	1/3	1/10
= 1	1	1
<= 0	1/3	1/10

This is DB2‘s documented estimation to estimate range predicate filter factors, if there are no literal values available.

Application Profiles

- The Idea
  - Application Dependent Optimization Profiles
    - Example1: “Access Path Stability” profile vs. “Access Path Peak Performance” profile
  - Compare with mobile phone profiles!
  - Compare with normal, sport, sport plus car driving options

“Normally, Dynamic Stability Control (DSC) helps to deliver optimal traction; but with M drive, DSC can also be set to track oriented M Dynamic Mode (MDM), which allows for more wheel slip and yaw angle. Optional Electronic Damping Control (EDC) can be set to comfort, normal or sport, depending upon your preferred handling settings and the surface of the road. Also engine response can be tweaked via a button in the center console which calls up normal, sport or sport plus programs. “

This statement is taken from BMW’s website to emphasize the tuning options of their M3 sports car.

A similar idea is to have different “*optimization profiles*” for different *types of applications*, for example an “*access path stability*” profile without the latest catalog statistics enhancements versus an “*access path peak performance*” profile including all options such as REOPT(AUTO), DATAREPEATFACTORF, up-to-date statistics without any additional updates, Zparm NPGTHRSH=0, etc.

Application Profiles

- Different Sets of Statistics for Different Applications
  - Static Queries
    - Aggressive, Current Statistics
    - Zparm NPGTHRSH = 0 (or small positive value)
    - Reinstall before rebind
  - Dynamic Queries
    - Balanced NLEVELS
    - CLUSTERRATIOF = 0.95
    - DATAREPEATFACTORF = -1
    - Zparm NPGTHRSH = 2000000 (V8: -1)

Application Profiles

- Daily Business with Application Profiles
  - Switch to *peak performance profile* (correct statistics) before rebind
    - Update from SYSINDEXES_HIST
    - SET SYSPARM LOAD(name)
  - Switch to *access path stability profile*
    - After Runstats
    - After (Re)bind
    - Don't forget to empty the cache

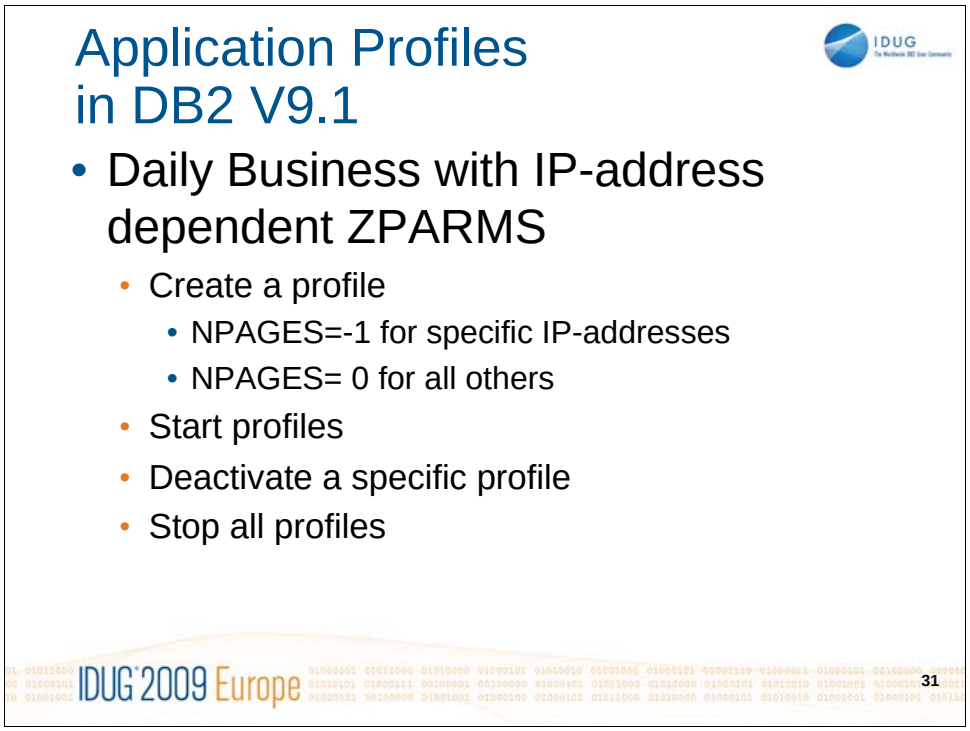
Another statement from BMW: „Many DSC (dynamic stability control) systems haven an „off" override switch so the driver can disable DSC, which may be desirable when badly stuck in mud or snow, or driving on a beach.However, DSC defaults to „on" when the ignition is re-started. Some DSC systems that lack an „off switch" can be temporarily disabled through an undocumented series of brake pedal and handbrake operations".

At least, this sounds not unfamiliar to DB2.

Application Profiles in DB2 V9.1

- IP-address dependent ZPARMS
  - NPAGES Threshold
    - -1:Use index access for as many tables as possible.
    - 0:Select access path based on cost.
    - n:Use index access on tables with NPAGES < n. Select AP based on cost for tabels with > n pages.
  - Enable/Disable Star Joins
    - DISABLE: do not use star join processing
    - ENABLE: use star join processing when possible.

Look into DB2 V9.1 Administration Guide, chapter 24 „Monitoring SQL performance“, section „Monitoring and optimizing queries with profile tables“ for further details and explanations.



The new DB2 z/OS V9.1 profile tables are designed to be used by optimization service center (OSC), and in order to highlight queries (statically and dynamically bound) which reach user-defined thresholds such as CPU limits. Additionally, these tables might contain IP-address specific (or plan/package based) modifications to subsystem parameters.

To activate this functionality (in DB2 z/OS V9.1 NFM only):

1. Create table SYSIBM.DSN_PROFILE_TABLE
2. Create table SYSIBM.DSN_PROFILE_HISTORY
3. Create table SYSIBM.DSN_PROFILE_ATTRIBUTES
4. Create table SYSIBM.DSN_PROFILE_ATTRIBUTES_HISTORY
5. Create a profile for the statements you want to assign new subsystem parameters, and set PROFILE_ENABLED to ‚Y‘.
6. Start the profile by executing –STA PROFILE command.
7. Deactivate this functionality by either
   7a. Issuing the STOP PROFILE command (to stop all profiles)
   7b. Setting PROFILE_ENABLED=‚N‘ for a specific profile

Summary

- V8 → V9 CM Migration
  - Prepare for Access Path Fallback
- Limit Exposure to Access Path Instability
  - Provide Clear Access Path Choices to DB2
- Dynamic Query Access Path Fallback
  - Be Prepared for Catalog Updates
- New Approach: Application Profiles

Where *query tuning* is likely to be considered as a part of performance management, the art of *access path stability* is a discipline of high availability management.

Session: F04

Access Path Stability for Dynamic Queries

Thomas Baumann

Swiss Mobiliar, Switzerland

thomas.baumann@mobi.ch

Since 1992, Thomas has been focusing on understanding how the DB2 database engine works. He has a master degree of computer sciences from ETH Zurich, Switzerland, and is currently working as data management team leader at Swiss Mobiliar Insurance in Berne, Switzerland. If he is not in his office trying how to get the most out of DB2, he is somewhere lecturing on DB2 optimization, Thomas is a member of the IDUG speakers hall of fame, and an internationally recognized expert on database auditing..