May 6–10, 2007

San Jose Convention Center

San Jose, California, USA

IDUG® 2007

North America

Session: B09

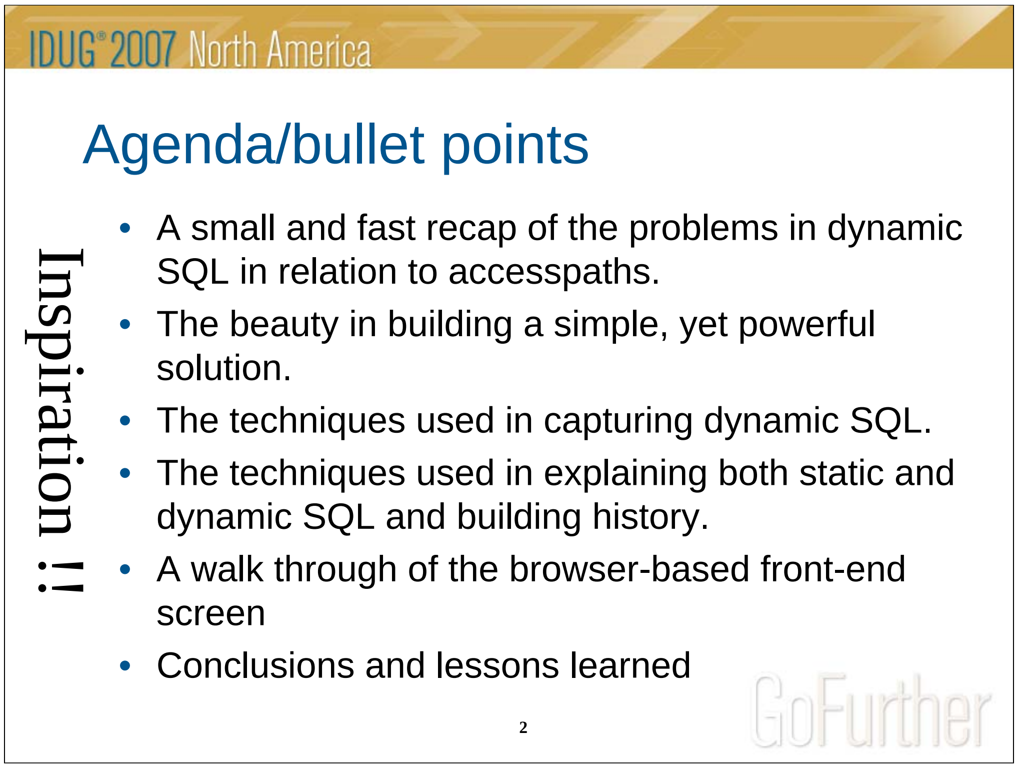
SQL and Access Paths - "A DIY way to survive" - for both static and dynamic SQL

Frank Petersen
Bankdata, Denmark

May 09, 2007 9:50 a.m. – 10:50 a.m.

Platform: DB2 for Z/os

GoFurther


IDUG® 2007 North America

Agenda/bullet points

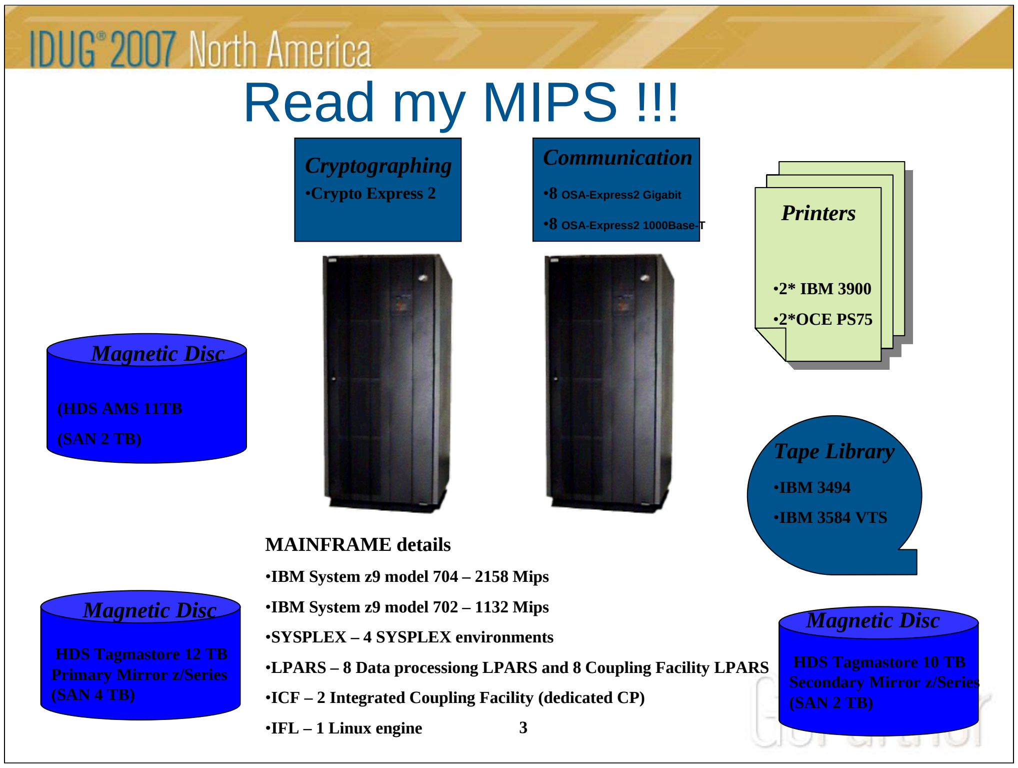
Inspiration !!

- A small and fast recap of the problems in dynamic SQL in relation to accesspaths.
- The beauty in building a simple, yet powerful solution.
- The techniques used in capturing dynamic SQL.
- The techniques used in explaining both static and dynamic SQL and building history.
- A walk through of the browser-based front-end screen
- Conclusions and lessons learned

GoFurther

2

This presentation will look into a practical implementation of some ideas shown at IDUG, Berlin 2005



IDUG® 2007 North America

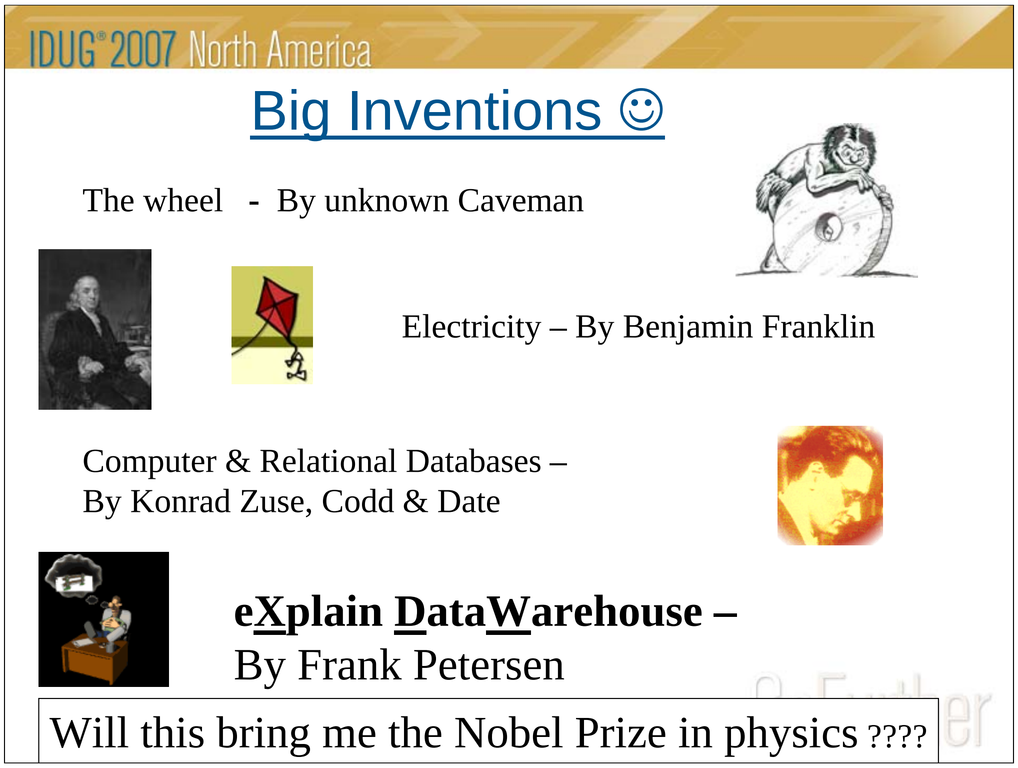
Read my MIPS !!!

MAINFRAME details

- **IBM System z9 model 704 – 2158 Mips**
- **IBM System z9 model 702 – 1132 Mips**
- **SYSPLEX – 4 SYSPLEX environments**
- **LPARS – 8 Data processiong LPARS and 8 Coupling Facility LPARS**
- **ICF – 2 Integrated Coupling Facility (dedicated CP)**
- **IFL – 1 Linux engine**

3

Some boxes with different sizes, but always toooo small !!!!



IDUG® 2007 North America

Big Inventions ☺

The wheel - By unknown Caveman

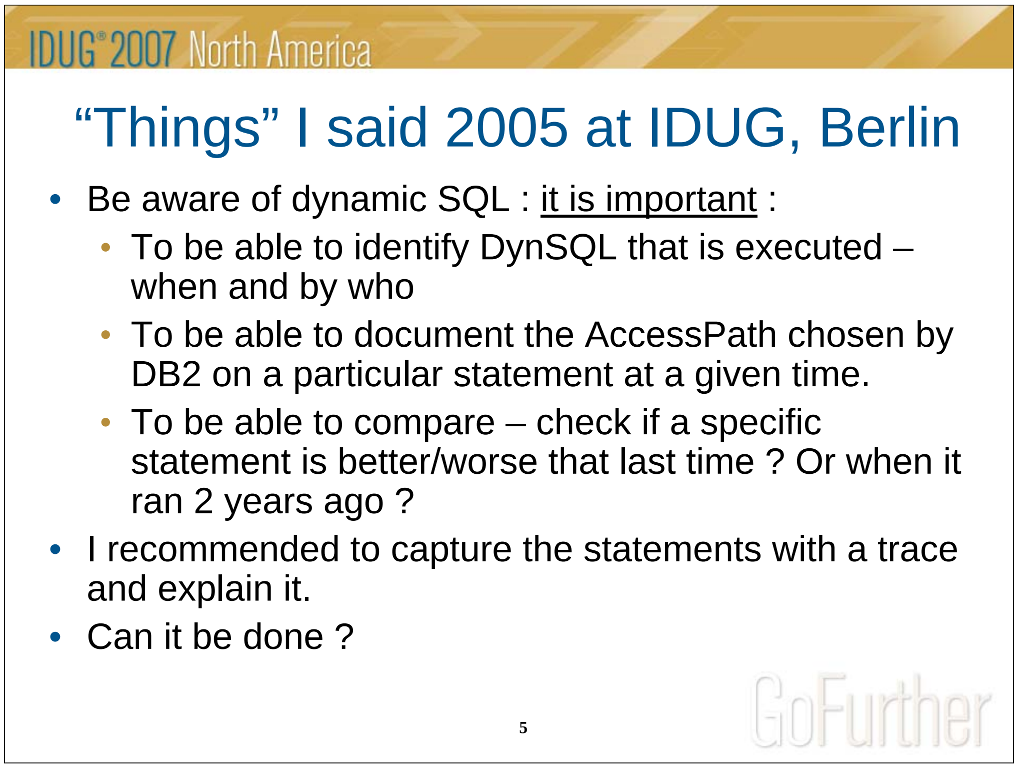
Electricity – By Benjamin Franklin

Computer & Relational Databases –
By Konrad Zuse, Codd & Date

e<u>X</u>plain <u>D</u>ata<u>W</u>arehouse –
By Frank Petersen

Will this bring me the Nobel Prize in physics ????

Some boxes with different sizes, but always toooo small !!!!


IDUG® 2007 North America

"Things" I said 2005 at IDUG, Berlin

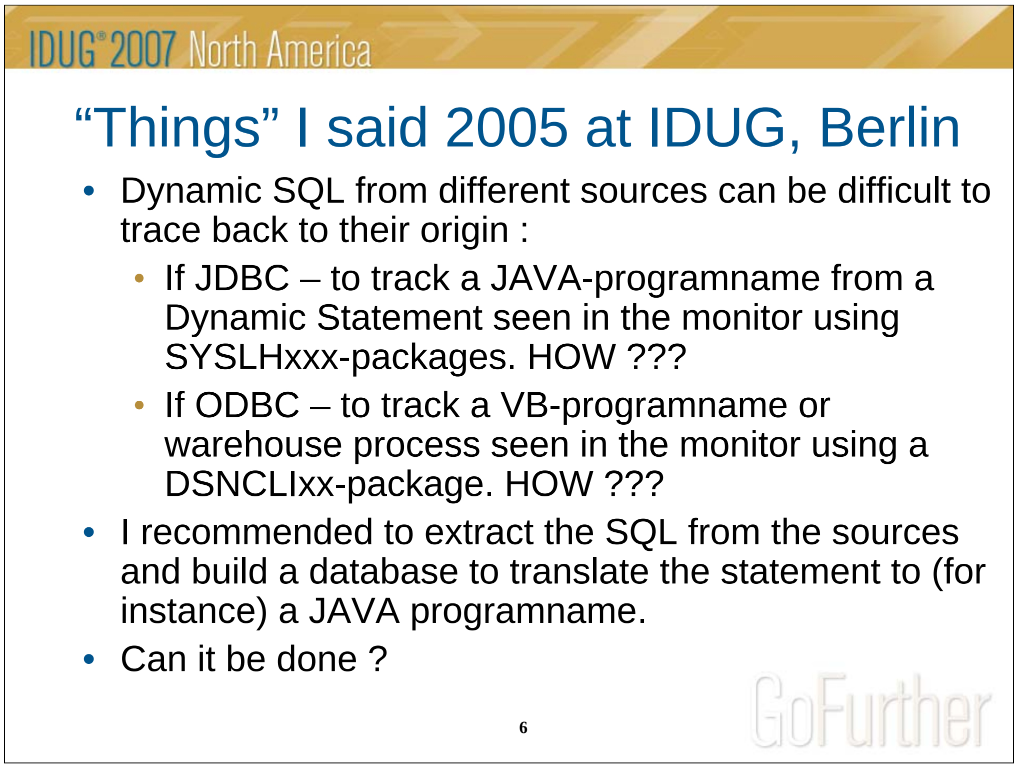
- Be aware of dynamic SQL : <u>it is important</u> :
  - To be able to identify DynSQL that is executed – when and by who
  - To be able to document the AccessPath chosen by DB2 on a particular statement at a given time.
  - To be able to compare – check if a specific statement is better/worse that last time ? Or when it ran 2 years ago ?
- I recommended to capture the statements with a trace and explain it.
- Can it be done ?

5

GoFurther

Last year I recommended several things. Here we will concentrate on 2 issues.

Firstly the idea of tracing all Dynamic SQL, explain it and save the result for viewing and comparing.



IDUG® 2007 North America

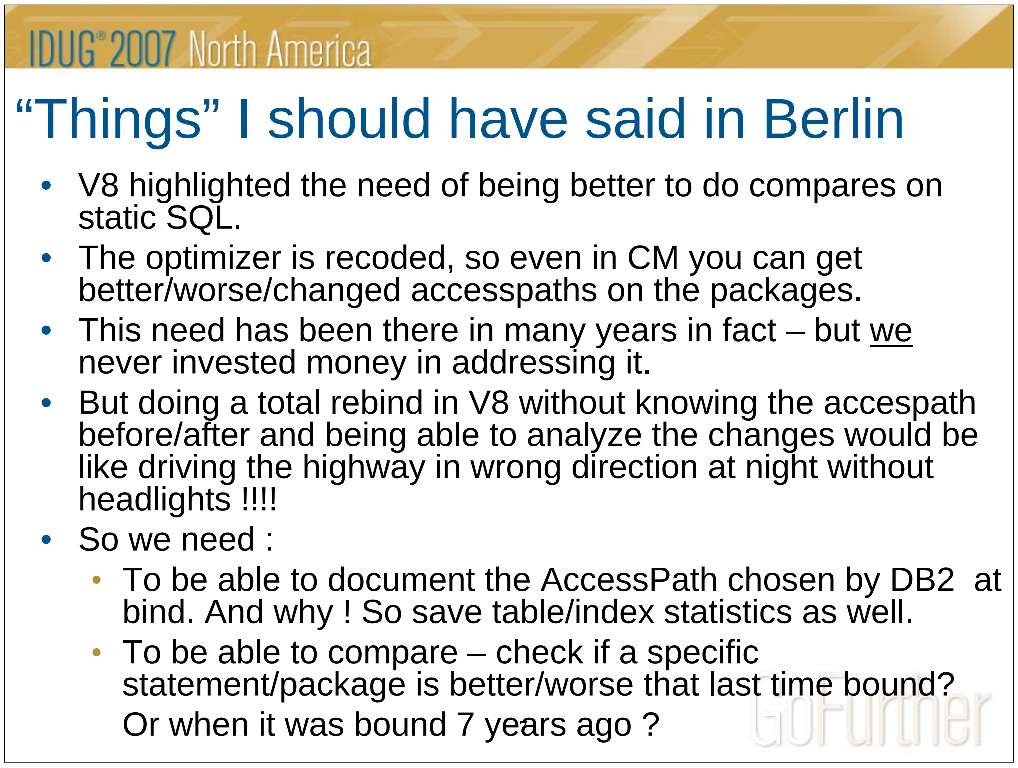
“Things” I said 2005 at IDUG, Berlin

- Dynamic SQL from different sources can be difficult to trace back to their origin :
  - If JDBC – to track a JAVA-programname from a Dynamic Statement seen in the monitor using SYSLHxxx-packages. HOW ???
  - If ODBC – to track a VB-programname or warehouse process seen in the monitor using a DSNCLIxx-package. HOW ???
- I recommended to extract the SQL from the sources and build a database to translate the statement to (for instance) a JAVA programname.
- Can it be done ?

6

GoFurther

Secondly I said that one could extract the Dynamic SQL text from the different sources; JAVA sources, ETL-processes, ODBC programs etc. and put it into some sort of repository. If we trace the dynamic SQL we should be able find it in the repository and in this way track down the JAVA-program, the MS-program or the WareHouseProcess responable.



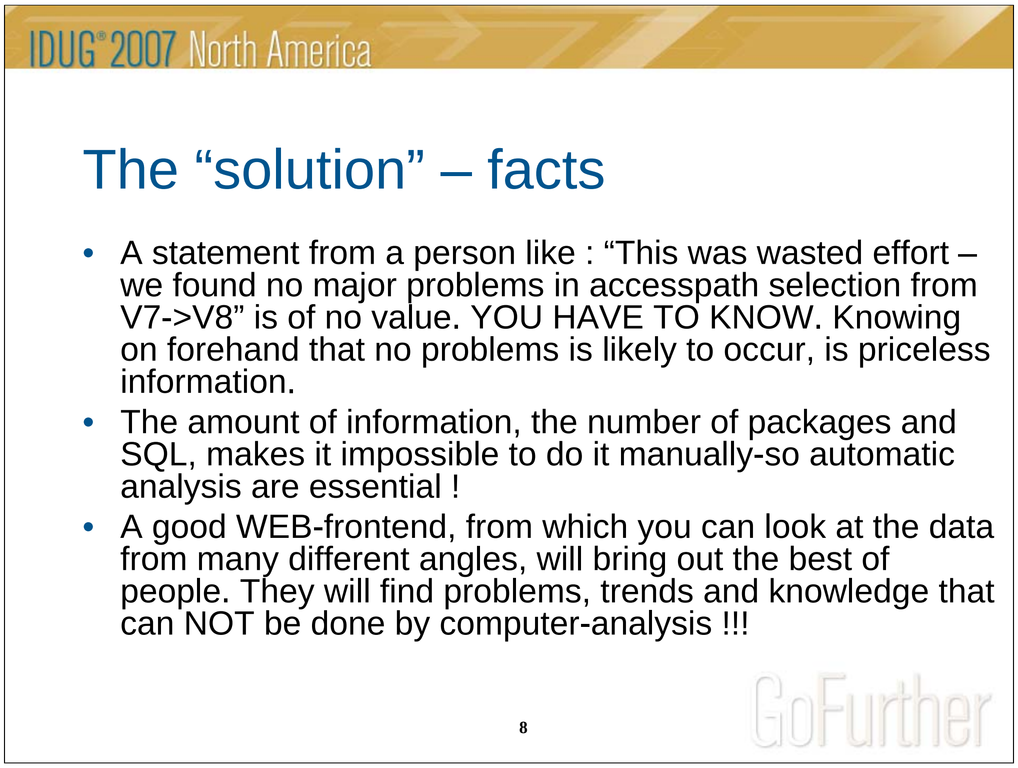
IDUG® 2007 North America

“Things” I should have said in Berlin

- V8 highlighted the need of being better to do compares on static SQL.
- The optimizer is recoded, so even in CM you can get better/worse/changed accesspaths on the packages.
- This need has been there in many years in fact – but we never invested money in addressing it.
- But doing a total rebind in V8 without knowing the accespath before/after and being able to analyze the changes would be like driving the highway in wrong direction at night without headlights !!!!
- So we need :
  - To be able to document the AccessPath chosen by DB2 at bind. And why ! So save table/index statistics as well.
  - To be able to compare – check if a specific statement/package is better/worse that last time bound? Or when it was bound 7 years ago ?

GoFurther

Since last year it became obviously that there was the same need for explaining and saving an explain-report in words and do version comparing for static SQL as well.

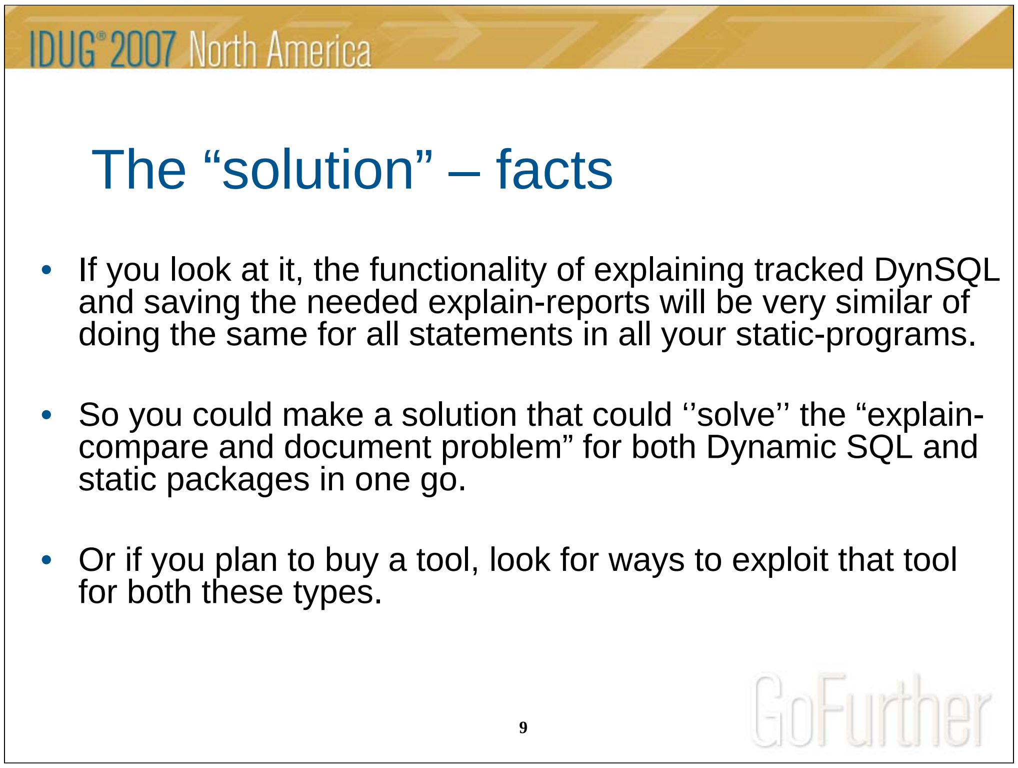


The “solution” – facts

- A statement from a person like : “This was wasted effort – we found no major problems in accesspath selection from V7->V8” is of no value. YOU HAVE TO KNOW. Knowing on forehand that no problems is likely to occur, is priceless information.
- The amount of information, the number of packages and SQL, makes it impossible to do it manually-so automatic analysis are essential !
- A good WEB-frontend, from which you can look at the data from many different angles, will bring out the best of people. They will find problems, trends and knowledge that can NOT be done by computer-analysis !!!

GoFurther

Some installation go to NF without a worry. We did not have that desire. We would love to be able to see on forehand that we should not expect any problems. So that was what was our major objective !!!


IDUG® 2007 North America

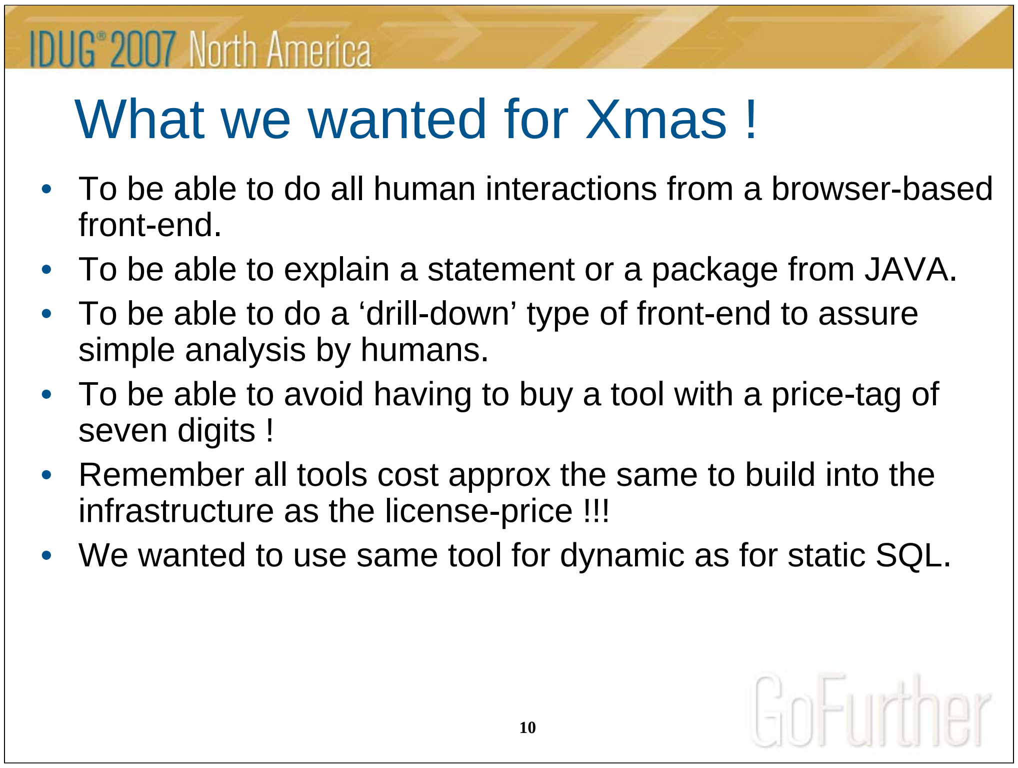
The “solution” – facts

- If you look at it, the functionality of explaining tracked DynSQL and saving the needed explain-reports will be very similar of doing the same for all statements in all your static-programs.
- So you could make a solution that could ‘’solve’’ the “explain-compare and document problem” for both Dynamic SQL and static packages in one go.
- Or if you plan to buy a tool, look for ways to exploit that tool for both these types.

9

GoFurther

It looked as if that part was easy to add on to a solution for dynamic SQL, if that could be build !!


IDUG® 2007 North America

What we wanted for Xmas !

- To be able to do all human interactions from a browser-based front-end.
- To be able to explain a statement or a package from JAVA.
- To be able to do a ‘drill-down’ type of front-end to assure simple analysis by humans.
- To be able to avoid having to buy a tool with a price-tag of seven digits !
- Remember all tools cost approx the same to build into the infrastructure as the license-price !!!
- We wanted to use same tool for dynamic as for static SQL.

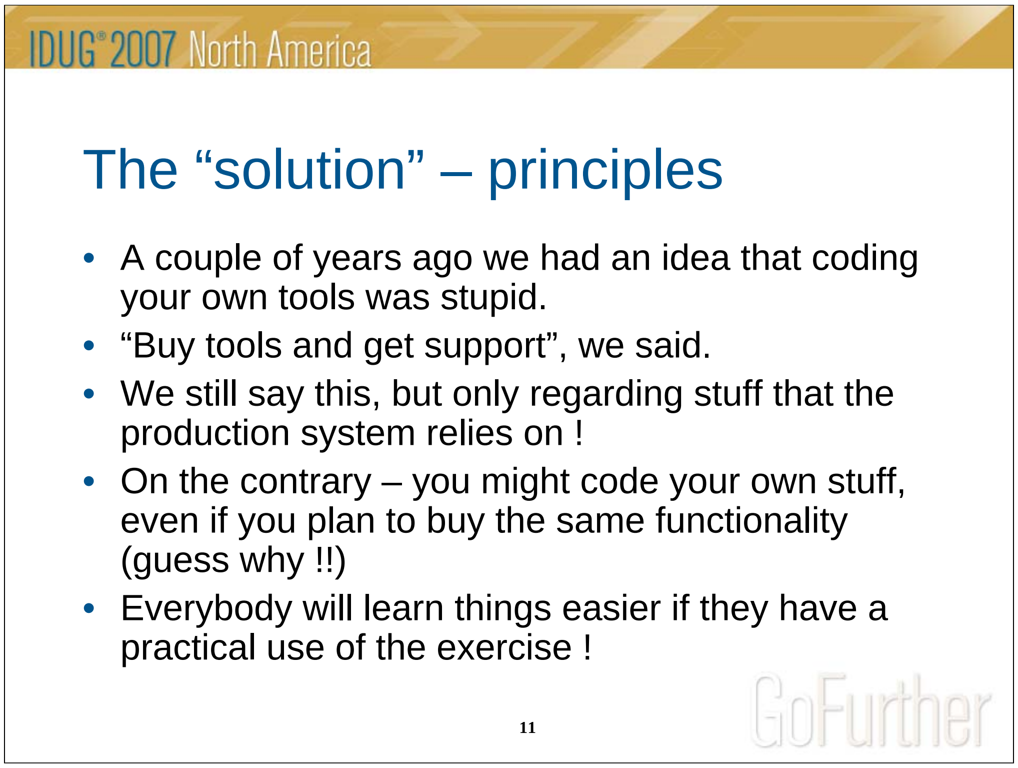
10

GoFurther

If we could/would build a solution, we had some ideas :

It should be driven by a WEB-browser as this tend s to get more people in as users. It could also push our JAVA-education a little further once again and it will automatically give a nice drill-through type approach.

The thing driving us, was money. We have none and we often feel that we only use expensive software to a small degree. And we pay for the whole package and even the wrapping !!! We also sometimes feel we pay for maintenance and do you never get the feeling that that is wasted money ???


IDUG® 2007 North America

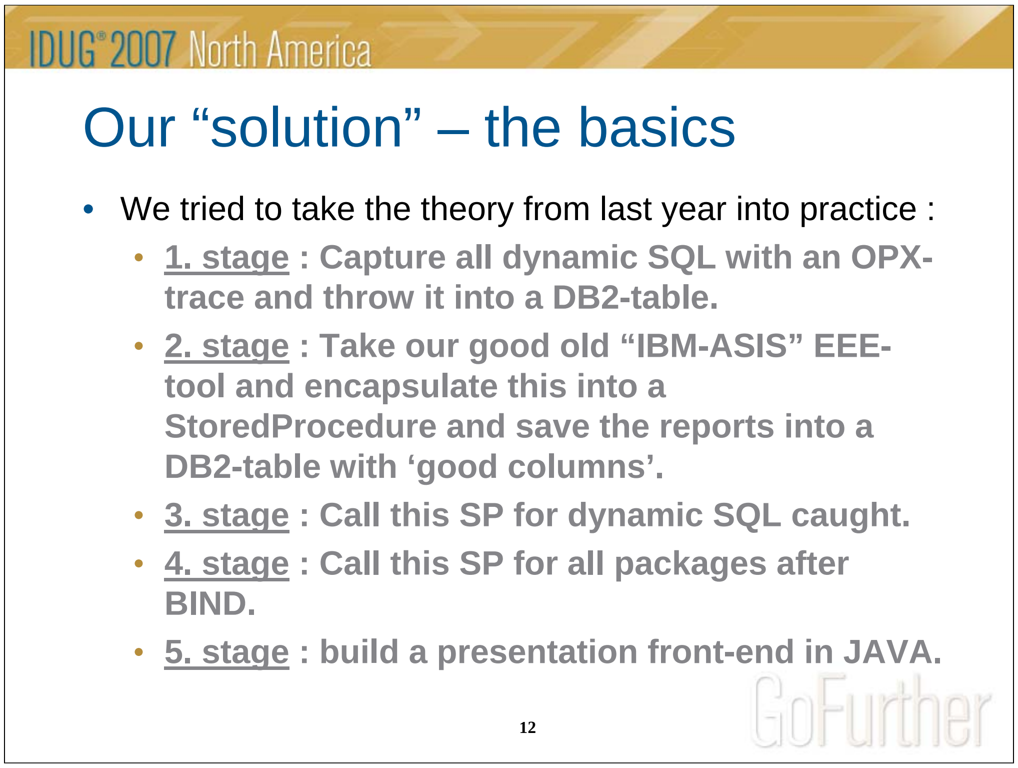
The “solution” – principles

- A couple of years ago we had an idea that coding your own tools was stupid.
- “Buy tools and get support”, we said.
- We still say this, but only regarding stuff that the production system relies on !
- On the contrary – you might code your own stuff, even if you plan to buy the same functionality (guess why !!)
- Everybody will learn things easier if they have a practical use of the exercise !

11

GoFurther

In our experience, home-build tools that have no relations to the production systems ability to run, can be an extremely good investment. Often we have had a very bad experience when recalculating how much money have been invested in getting an expensive tool to run and getting it implemented into our infrastructure. We have seen here that the cost of building a tool is probably less that even implementing a bought-in tool.


IDUG® 2007 North America

Our “solution” – the basics

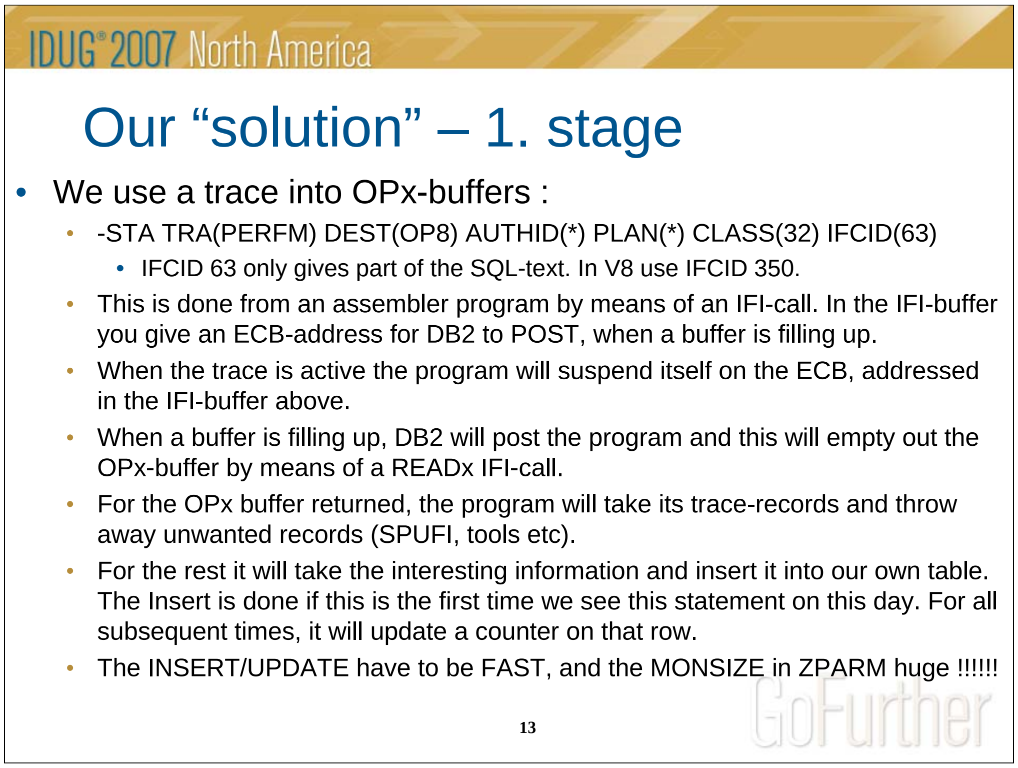
- We tried to take the theory from last year into practice :
  - 1. stage : Capture all dynamic SQL with an OPX-trace and throw it into a DB2-table.
  - 2. stage : Take our good old “IBM-ASIS” EEE-tool and encapsulate this into a StoredProcedure and save the reports into a DB2-table with ‘good columns’.
  - 3. stage : Call this SP for dynamic SQL caught.
  - 4. stage : Call this SP for all packages after BIND.
  - 5. stage : build a presentation front-end in JAVA.

12

GoFurther

The solution regarding explaining and comparing SQL could be taken in smaller steps, where each steps could in fact be useful. That is always a good approach.

First step : Let’s trace the Dynamic SQL !


IDUG® 2007 North America

Our "solution" – 1. stage

- We use a trace into OPx-buffers :
  - -STA TRA(PERFM) DEST(OP8) AUTHID(*) PLAN(*) CLASS(32) IFCID(63)
    - IFCID 63 only gives part of the SQL-text. In V8 use IFCID 350.
  - This is done from an assembler program by means of an IFI-call. In the IFI-buffer you give an ECB-address for DB2 to POST, when a buffer is filling up.
  - When the trace is active the program will suspend itself on the ECB, addressed in the IFI-buffer above.
  - When a buffer is filling up, DB2 will post the program and this will empty out the OPx-buffer by means of a READx IFI-call.
  - For the OPx buffer returned, the program will take its trace-records and throw away unwanted records (SPUFI, tools etc).
  - For the rest it will take the interesting information and insert it into our own table. The Insert is done if this is the first time we see this statement on this day. For all subsequent times, it will update a counter on that row.
  - The INSERT/UPDATE have to be FAST, and the MONSIZE in ZPARM huge !!!!!

13

GoFurther

We are in the wonderful world of the assembler language !!!! We build a program that will take several input parameters : USER to trace , PLAN to trace , PACKAGES to trace and how many minutes to trace.

-

From these parameters it will construct a STA TRACE and execute this, using the instrumentation facility.

The OPx buffers are just wonderfull. When the STA TRACE is executed it will give an ECB-address for DB2 to POST when a OPx-buffer is full. So after starting the trace the assembler program will just go into a WAIT on that ECB-address.

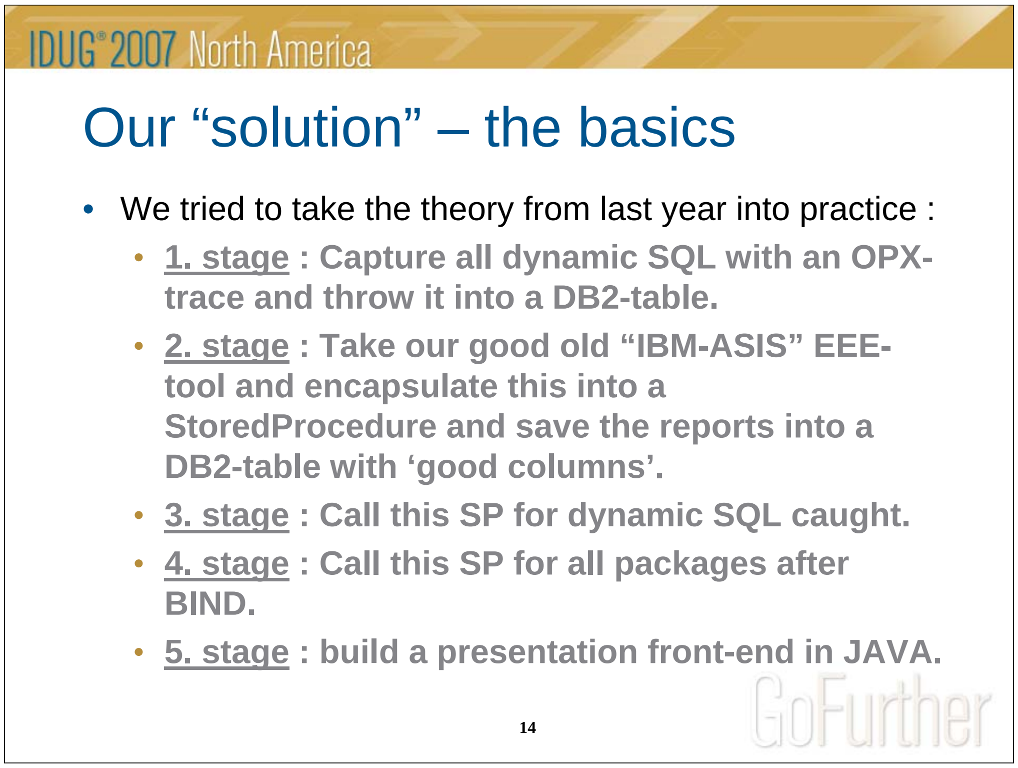
When DB2 POSTs the program it will take its OPx-buffer, find the appropriate records, transform and insert these in our own DB2-table. However it will only insert the same statement one time per day. If the statement reappears we will just update a counter.

When there are no more records in the buffers it will go back in a WAIT.

-

Once the number of minutes have ticked away it will stop its own trace and terminate.

It is as simple as that !!!!!!


IDUG® 2007 North America

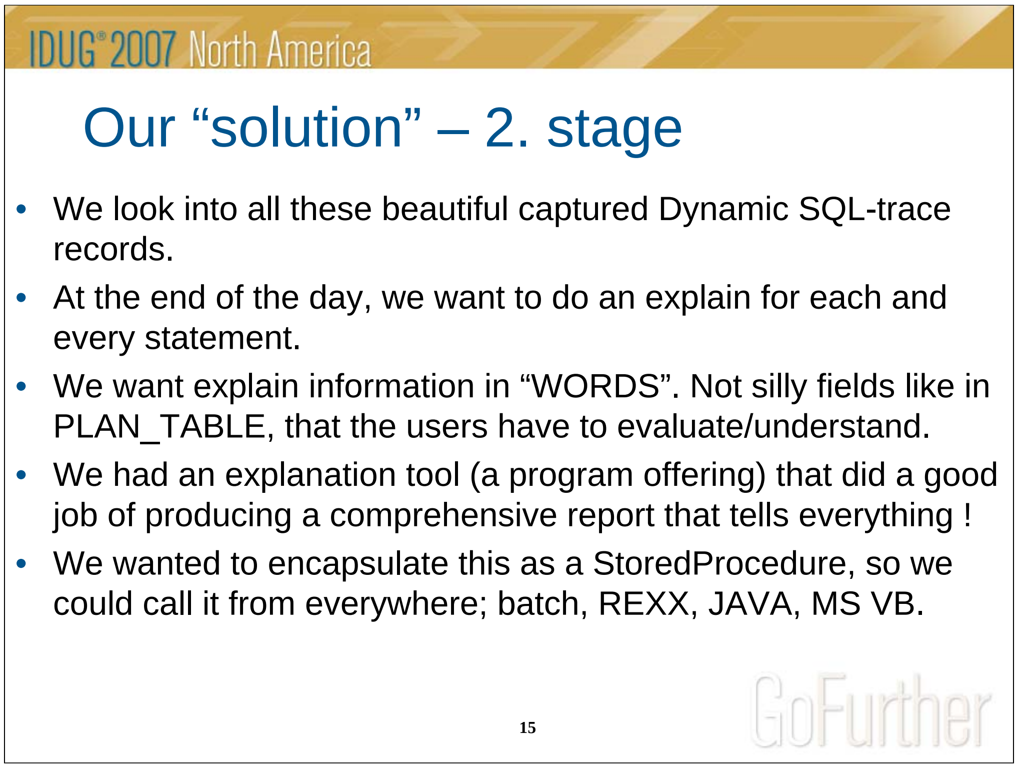
Our "solution" – the basics

- We tried to take the theory from last year into practice :
  - 1. stage : Capture all dynamic SQL with an OPX-trace and throw it into a DB2-table.
  - 2. stage : Take our good old "IBM-ASIS" EEE-tool and encapsulate this into a StoredProcedure and save the reports into a DB2-table with 'good columns'.
  - 3. stage : Call this SP for dynamic SQL caught.
  - 4. stage : Call this SP for all packages after BIND.
  - 5. stage : build a presentation front-end in JAVA.

14

GoFurther

Now we have a DB2-table full of nice records containing different information as user , date , connection-id etc. We had this idea of being able to call our explain-tool as a stored procedure, enabling us to call it from everywhere. So this was next stage.


IDUG® 2007 North America

Our “solution” – 2. stage

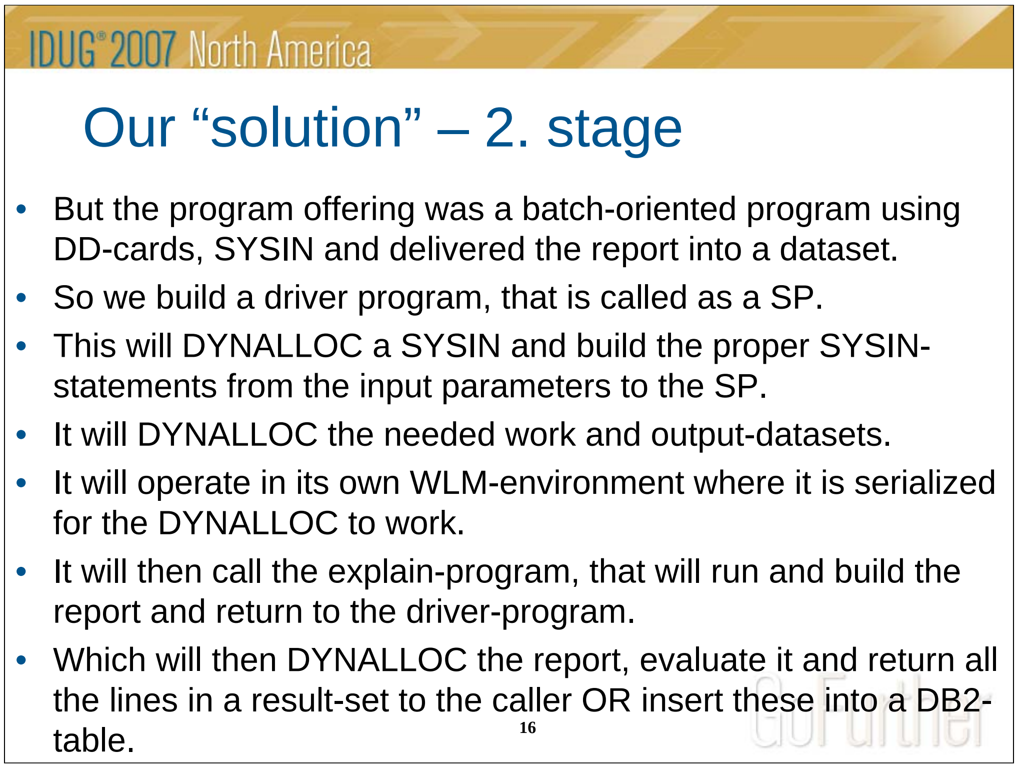
- We look into all these beautiful captured Dynamic SQL-trace records.
- At the end of the day, we want to do an explain for each and every statement.
- We want explain information in “WORDS”. Not silly fields like in PLAN_TABLE, that the users have to evaluate/understand.
- We had an explanation tool (a program offering) that did a good job of producing a comprehensive report that tells everything !
- We wanted to encapsulate this as a StoredProcedure, so we could call it from everywhere; batch, REXX, JAVA, MS VB.

15

GoFurther

For us it as important to save a report in words that are direct understandable. In this way we could reach a larger crowd !!

If we could call the tool as a SP, we could use it universally !!!



IDUG® 2007 North America

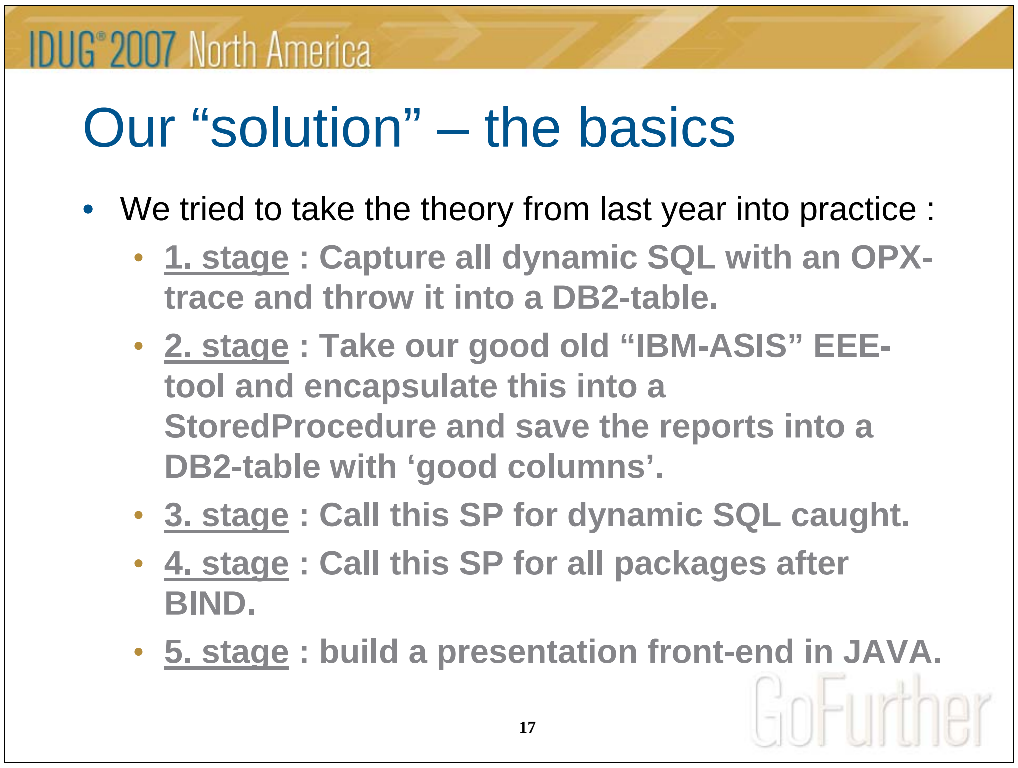
Our “solution” – 2. stage

- But the program offering was a batch-oriented program using DD-cards, SYSIN and delivered the report into a dataset.
- So we build a driver program, that is called as a SP.
- This will DYNALLOC a SYSIN and build the proper SYSIN-statements from the input parameters to the SP.
- It will DYNALLOC the needed work and output-datasets.
- It will operate in its own WLM-environment where it is serialized for the DYNALLOC to work.
- It will then call the explain-program, that will run and build the report and return to the driver-program.
- Which will then DYNALLOC the report, evaluate it and return all the lines in a result-set to the caller OR insert these into a DB2-table.

16

GoFurther

The big message here is that I have difficulties in finding anything that can not be encapsulated as a SP. Here we have a batch-program with a lot of DD-cards that was easily packaged with a couple of assembler-programs to do DYNALLOC and de-allocating again after the call. So I would dare to say that if you have an explain-tool, you can call it as a SP after 2 days work !!!!!!!!!



IDUG® 2007 North America

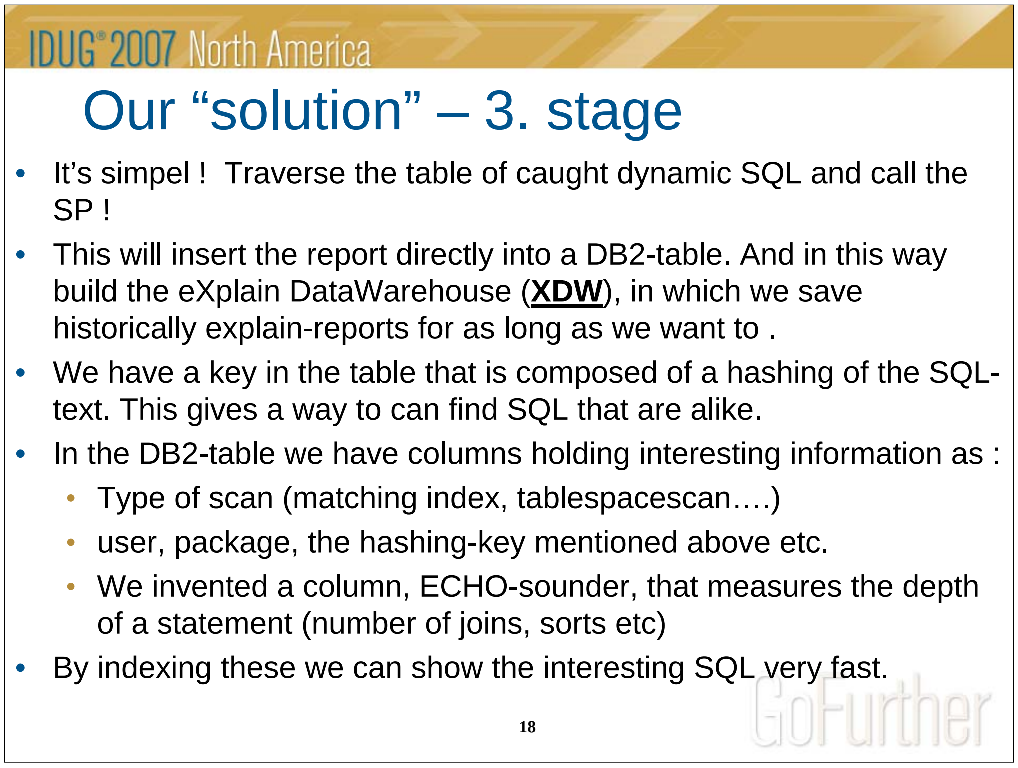
Our “solution” – the basics

- We tried to take the theory from last year into practice :
  - **1. stage : Capture all dynamic SQL with an OPX-trace and throw it into a DB2-table.**
  - **2. stage : Take our good old “IBM-ASIS” EEE-tool and encapsulate this into a StoredProcedure and save the reports into a DB2-table with ‘good columns’.**
  - **3. stage : Call this SP for dynamic SQL caught.**
  - **4. stage : Call this SP for all packages after BIND.**
  - **5. stage : build a presentation front-end in JAVA.**

17

GoFurther

Now we are ready to explain all the captured SQL


IDUG® 2007 North America

Our “solution” – 3. stage

- It’s simpel ! Traverse the table of caught dynamic SQL and call the SP !
- This will insert the report directly into a DB2-table. And in this way build the eXplain DataWarehouse (**XDW**), in which we save historically explain-reports for as long as we want to .
- We have a key in the table that is composed of a hashing of the SQL-text. This gives a way to can find SQL that are alike.
- In the DB2-table we have columns holding interesting information as :
  - Type of scan (matching index, tablespacescan….)
  - user, package, the hashing-key mentioned above etc.
  - We invented a column, ECHO-sounder, that measures the depth of a statement (number of joins, sorts etc)
- By indexing these we can show the interesting SQL very fast.

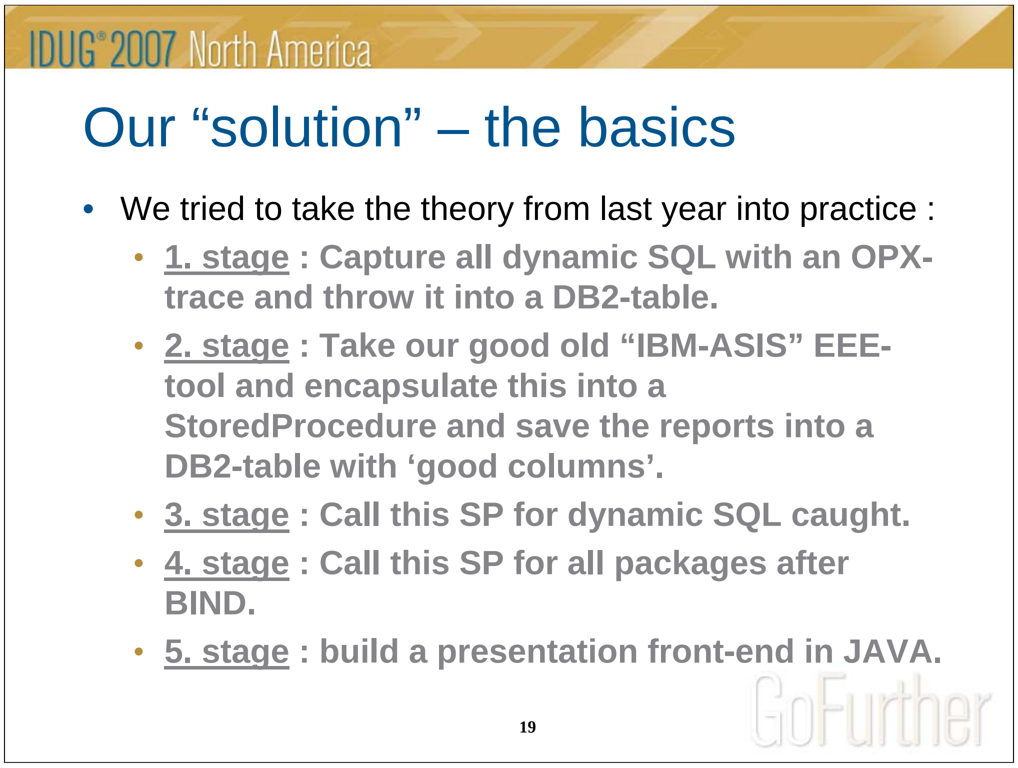
18

GoFurther

In the wrapping of the explain-tool as a SP we also introduced a parameter, so the SP can automatically insert all the report-lines in a DB2-table, thus making it possible in all future to find exactly that explain-report. The table will have columns of the accesspath, user, costs etc upon which we could build indexes for easy retrieval.

The parameter could also indicate that the report should be returned in a resultset, thus we could call a real-time explain from JAVA, just presenting the report on the browser.

The SQL-text was thrown through a hashing-calculation and the result used as keys in all tables involved. In this way it is easy to find SQL that is alike, over programs, versions and statements !!!


IDUG® 2007 North America

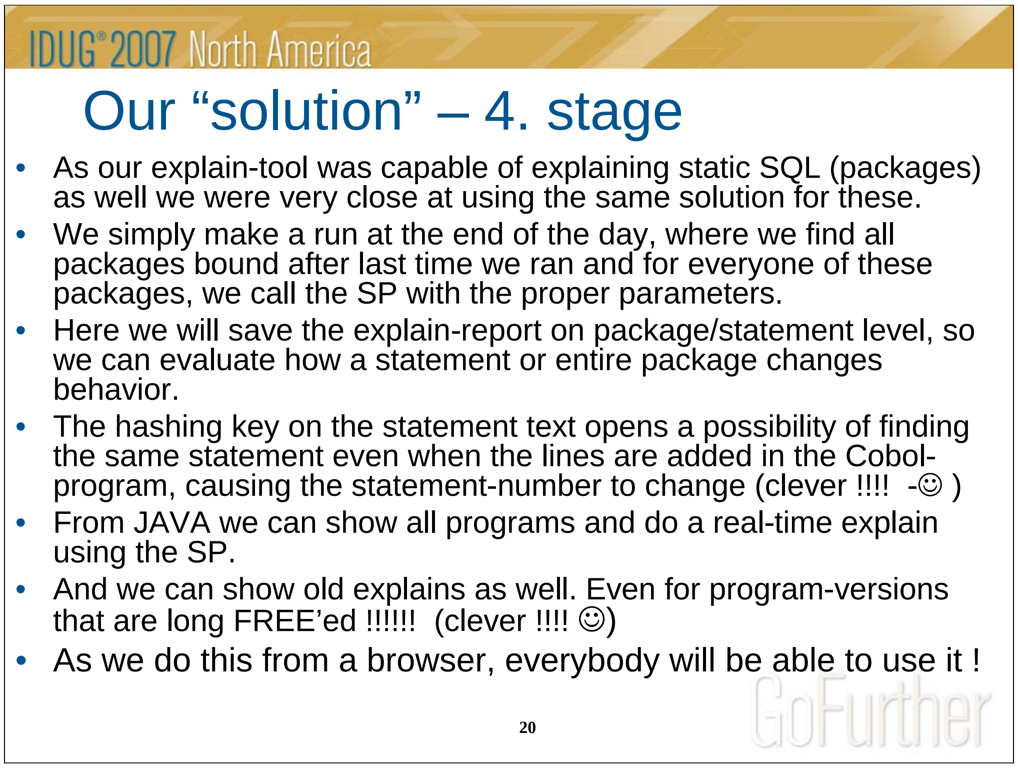
Our “solution” – the basics

- We tried to take the theory from last year into practice :
  - **1. stage : Capture all dynamic SQL with an OPX-trace and throw it into a DB2-table.**
  - **2. stage : Take our good old “IBM-ASIS” EEE-tool and encapsulate this into a StoredProcedure and save the reports into a DB2-table with ‘good columns’.**
  - **3. stage : Call this SP for dynamic SQL caught.**
  - **4. stage : Call this SP for all packages after BIND.**
  - **5. stage : build a presentation front-end in JAVA.**

19

GoFurther

Now we wanted to expand the solution to static SQL as well……


IDUG® 2007 North America

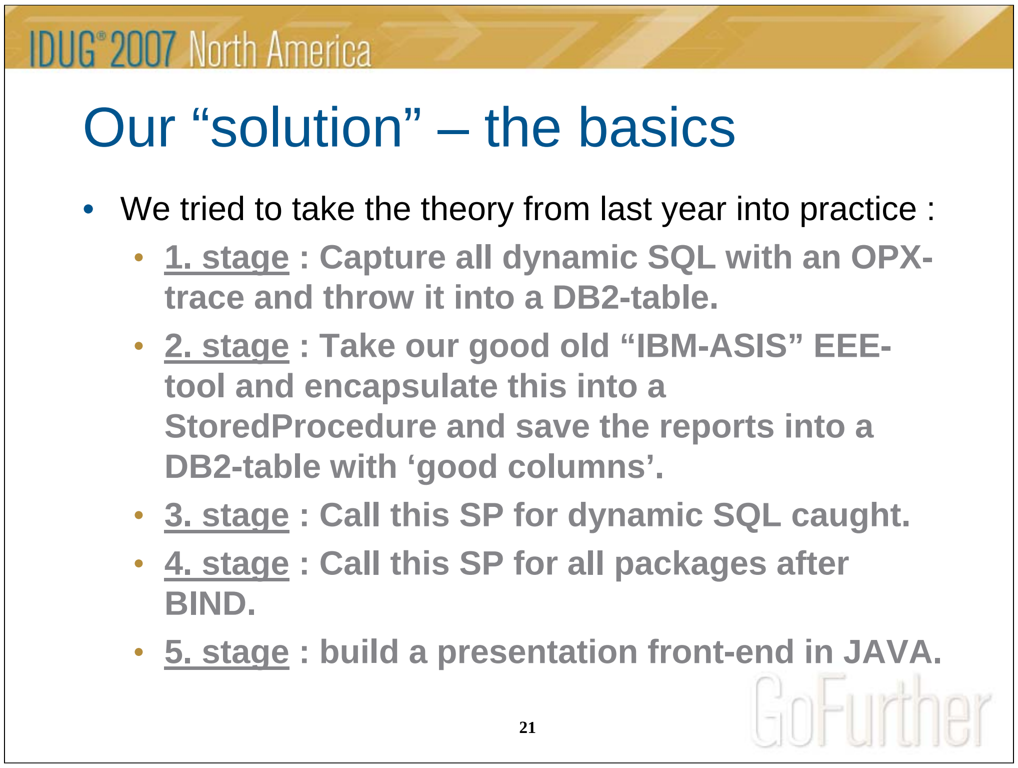
Our "solution" – 4. stage

- As our explain-tool was capable of explaining static SQL (packages) as well we were very close at using the same solution for these.
- We simply make a run at the end of the day, where we find all packages bound after last time we ran and for everyone of these packages, we call the SP with the proper parameters.
- Here we will save the explain-report on package/statement level, so we can evaluate how a statement or entire package changes behavior.
- The hashing key on the statement text opens a possibility of finding the same statement even when the lines are added in the Cobol-program, causing the statement-number to change (clever !!!! -☺)
- From JAVA we can show all programs and do a real-time explain using the SP.
- And we can show old explains as well. Even for program-versions that are long FREE'ed !!!!!! (clever !!!! ☺)
- As we do this from a browser, everybody will be able to use it !

20

GoFurther

A new table, of course, where we anchor all packages that we find. A join of this table with SYSIBM.SYSPACKAGES will easily find packages that was bound since last execution and now we just call the explain-tool for that package. It was soooo easy !!!


IDUG® 2007 North America

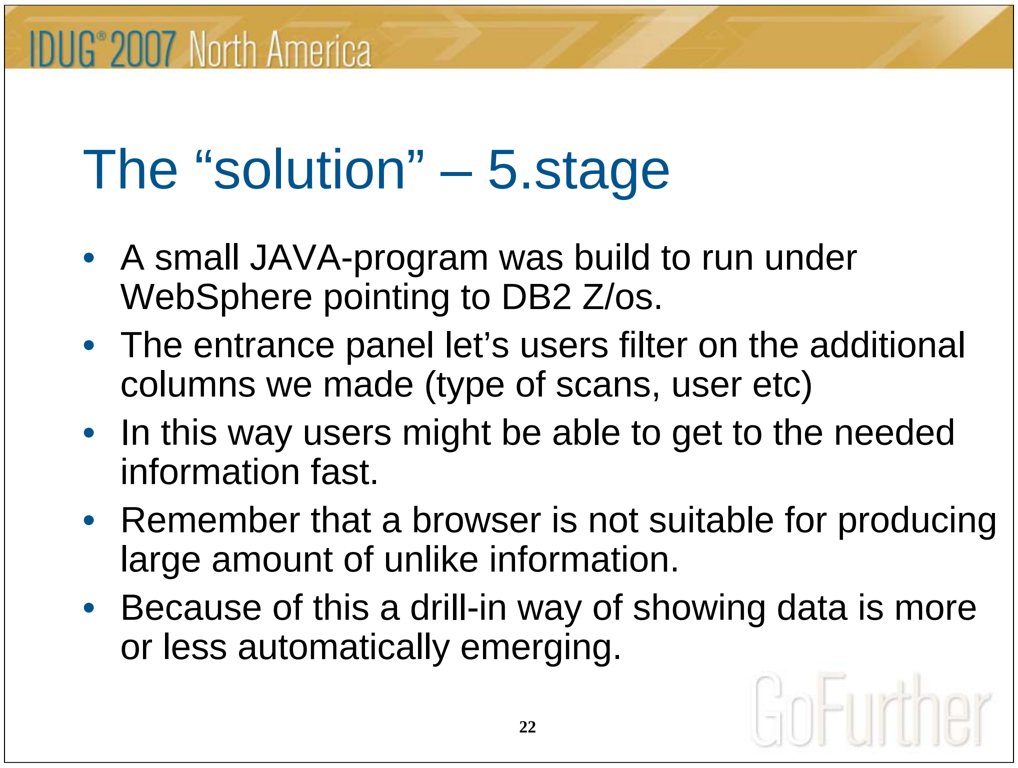
Our "solution" – the basics

- We tried to take the theory from last year into practice :
  - **1. stage : Capture all dynamic SQL with an OPX-trace and throw it into a DB2-table.**
  - **2. stage : Take our good old "IBM-ASIS" EEE-tool and encapsulate this into a StoredProcedure and save the reports into a DB2-table with 'good columns'.**
  - **3. stage : Call this SP for dynamic SQL caught.**
  - **4. stage : Call this SP for all packages after BIND.**
  - **5. stage : build a presentation front-end in JAVA.**

21

GoFurther

So now we just need a good front-end to be able to display all these wonderful data. JAVA, naturally ……



IDUG® 2007 North America

The "solution" – 5.stage

- A small JAVA-program was build to run under WebSphere pointing to DB2 Z/os.
- The entrance panel let's users filter on the additional columns we made (type of scans, user etc)
- In this way users might be able to get to the needed information fast.
- Remember that a browser is not suitable for producing large amount of unlike information.
- Because of this a drill-in way of showing data is more or less automatically emerging.

22

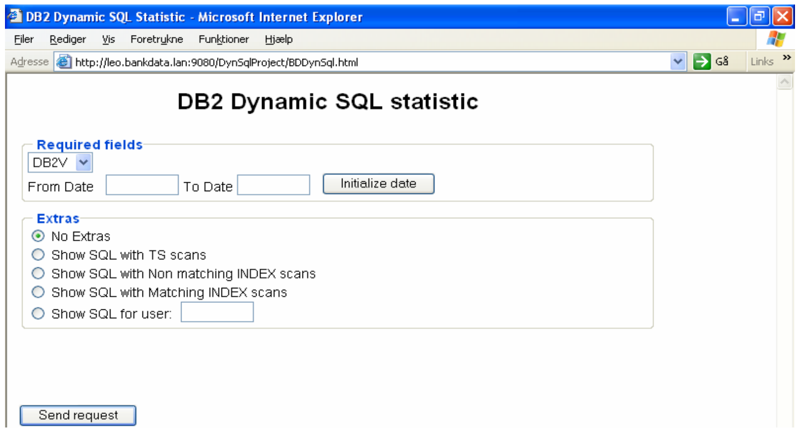
GoFurther

JAVA under WebSphere – that's state of the art , so go for it.

-

If you are not familiar with developing user interfaces, be aware that filtering and drill-in/down is the most important issue …….

The “solution” – 5. stage EXAMPLE


DB2 Dynamic SQL Statistic - Microsoft Internet Explorer

Filer Rediger Vis Foretrukne Funktioner Hjælp

Adresse http://leo.bankdata.lan:9080/DynSqlProject/BDDynSql.html

DB2 Dynamic SQL statistic

Required fields

DB2V

From Date ____ To Date ____ Initialize date

Extras

- [✓ No Extras]
- Show SQL with TS scans
- Show SQL with Non matching INDEX scans
- Show SQL with Matching INDEX scans
- Show SQL for user: ____

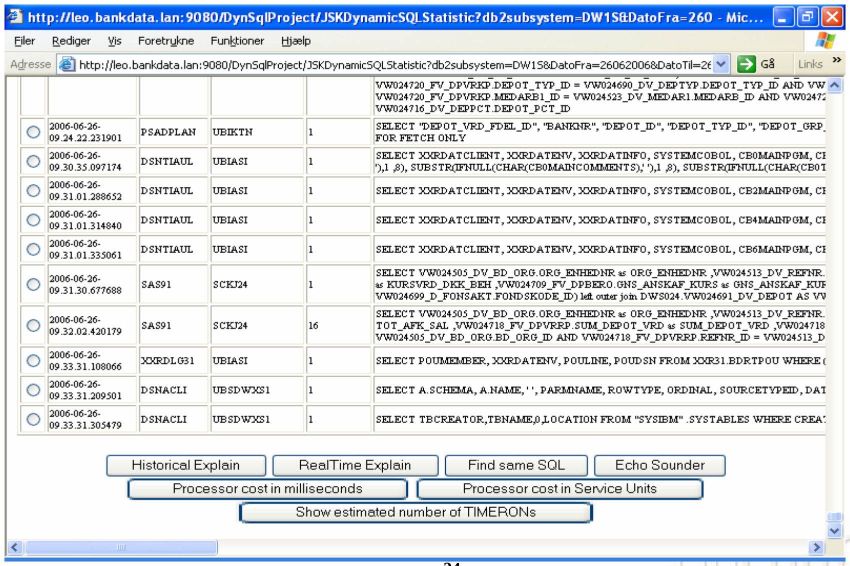
Send request

Just an example ! We could have included filtering on “processor cost” in milliseconds, service units and “estimated number of TIMERONS” as these were included as direct columns as well !

Our first-version frontend just include a couple of fields, but only imagination sets the limits !

IDUG® 2007 North America

Our “solution” – 5. stage EXAMPLE



http://leo.bankdata.lan:9080/DynSqlProject/JSKDynamicSQLStatistic?db2subsystem=DW1S&DatoFra=260 - Mic...

http://leo.bankdata.lan:9080/DynSqlProject/JSKDynamicSQLStatistic?db2subsystem=DW1S&DatoFra=26062006&DatoTil=26

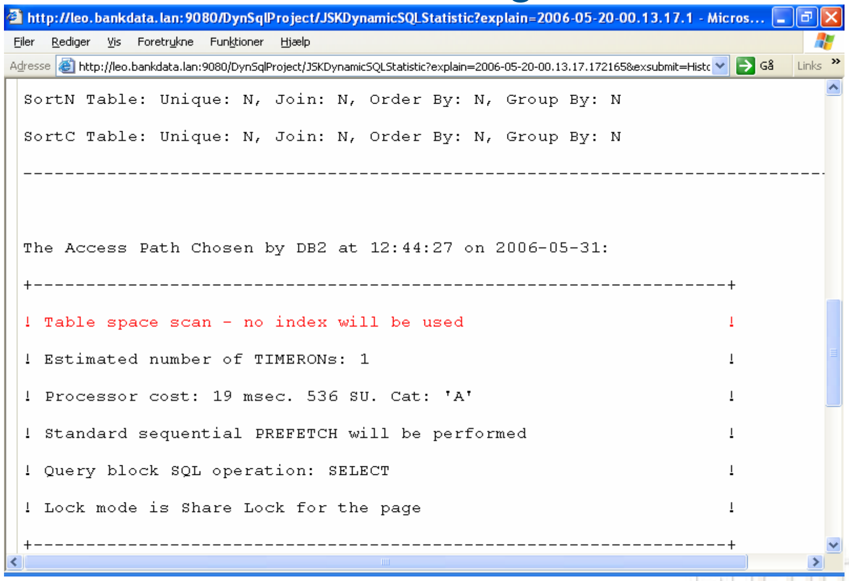
					VW024720_FV_DPVRKP.DEPOT_TYP_ID = VW024690_DV_DEPTYP.DEPOT_TYP_ID AND VW VW024720_FV_DPVRKP.MEDARB1_ID = VW024523_DV_MEDAR1.MEDARB_ID AND VW02472 VW024716_DV_DEPPCT.DEPOT_PCT_ID
○	2006-06-26- 09.24.22.231901	PSADPLAN	UBIKTN	1	SELECT "DEPOT_VRD_FDEL_ID", "BANKNR", "DEPOT_ID", "DEPOT_TYP_ID", "DEPOT_GRP_ FOR FETCH ONLY
○	2006-06-26- 09.30.35.097174	DSNTIAUL	UBIASI	1	SELECT XXRDATCLIENT, XXRDATENV, XXRDATINFO, SYSTEMCOBOL, CB0MAINPGM, CI '),1 ,8), SUBSTR(IFNULL(CHAR(CB0MAINCOMMENTS),' '),1 ,8), SUBSTR(IFNULL(CHAR(CB0T
○	2006-06-26- 09.31.01.288652	DSNTIAUL	UBIASI	1	SELECT XXRDATCLIENT, XXRDATENV, XXRDATINFO, SYSTEMCOBOL, CB2MAINPGM, CI
○	2006-06-26- 09.31.01.314840	DSNTIAUL	UBIASI	1	SELECT XXRDATCLIENT, XXRDATENV, XXRDATINFO, SYSTEMCOBOL, CB4MAINPGM, CI
○	2006-06-26- 09.31.01.335061	DSNTIAUL	UBIASI	1	SELECT XXRDATCLIENT, XXRDATENV, XXRDATINFO, SYSTEMCOBOL, CB6MAINPGM, CI
○	2006-06-26- 09.31.30.677688	SAS91	SCKJ24	1	SELECT VW024505_DV_BD_ORG.ORG_ENHEDNR as ORG_ENHEDNR ,VW024513_DV_REFNR. as KURSVRD_DKK_BEH ,VW024709_FV_DPBERO.GNS_ANSKAF_KURS as GNS_ANSKAF_KUR VW024699_D_FONSAKT.FONDSKODE_ID) left outer join DWS024.VW024691_DV_DEPOT AS VV
○	2006-06-26- 09.32.02.420179	SAS91	SCKJ24	16	SELECT VW024505_DV_BD_ORG.ORG_ENHEDNR as ORG_ENHEDNR ,VW024513_DV_REFNR. TOT_AFK_SAL ,VW024718_FV_DPVRRP.SUM_DEPOT_VRD as SUM_DEPOT_VRD ,VW024718 VW024505_DV_BD_ORG.BD_ORG_ID AND VW024718_FV_DPVRRP.REFNR_ID = VW024513_D
○	2006-06-26- 09.33.31.108066	XXRDLG31	UBIASI	1	SELECT POUMEMBER, XXRDATENV, POULINE, POUDSN FROM XXR31.BDRTPOU WHERE (
○	2006-06-26- 09.33.31.209501	DSNACLI	UBSDWXS1	1	SELECT A.SCHEMA, A.NAME, ' ', PARMNAME, ROWTYPE, ORDINAL, SOURCETYPEID, DAT
○	2006-06-26- 09.33.31.305479	DSNACLI	UBSDWXS1	1	SELECT TBCREATOR,TBNAME,0,LOCATION FROM "SYSIBM" .SYSTABLES WHERE CREA

24

Here we have picked all Dynamic SQL statements. Observe the counter and the text.

IDUG® 2007 North America

Our “solution” – 5. stage EXAMPLE


http://leo.bankdata.lan:9080/DynSqlProject/JSKDynamicSQLStatistic?explain=2006-05-20-00.13.17.1 - Micros...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKDynamicSQLStatistic?explain=2006-05-20-00.13.17.172165&exsubmit=Histc
Gå
Links

```
SortN Table: Unique: N, Join: N, Order By: N, Group By: N

SortC Table: Unique: N, Join: N, Order By: N, Group By: N

-----------------------------------------------------------------------------


The Access Path Chosen by DB2 at 12:44:27 on 2006-05-31:

+-------------------------------------------------------------------+

! Table space scan - no index will be used                          !

! Estimated number of TIMERONs: 1                                   !

! Processor cost: 19 msec. 536 SU. Cat: 'A'                         !

! Standard sequential PREFETCH will be performed                    !

! Query block SQL operation: SELECT                                 !

! Lock mode is Share Lock for the page                              !

+-------------------------------------------------------------------+
```

25

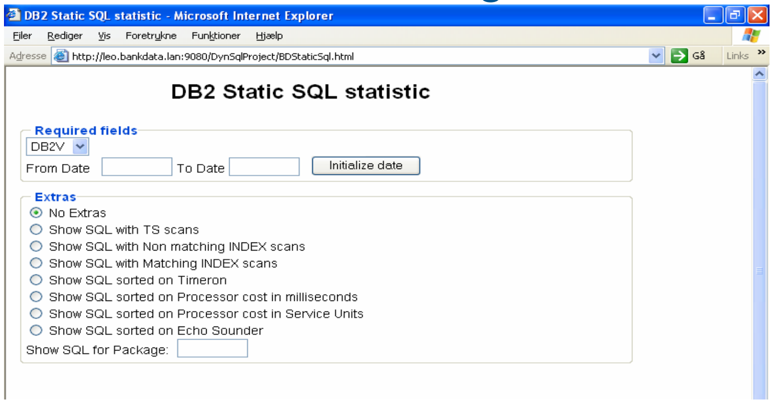
Go further

If the users picks a statement and click on ‘historical explain’, we will show the full explain-report as it was produced at the day the SQL was traced. No matter how long time ago. So here you could have YOUR favorite explain-report shown !!

Please notice on the previous foil that we can produce a ‘realtime’ report. This will show how things would look if the statement was prepared now !!

IDUG® 2007 North America

The “solution” – 5. stage EXAMPLE


DB2 Static SQL statistic - Microsoft Internet Explorer
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/BDStaticSql.html
Gå Links

DB2 Static SQL statistic

Required fields

DB2V

From Date ____ To Date ____ Initialize date

Extras

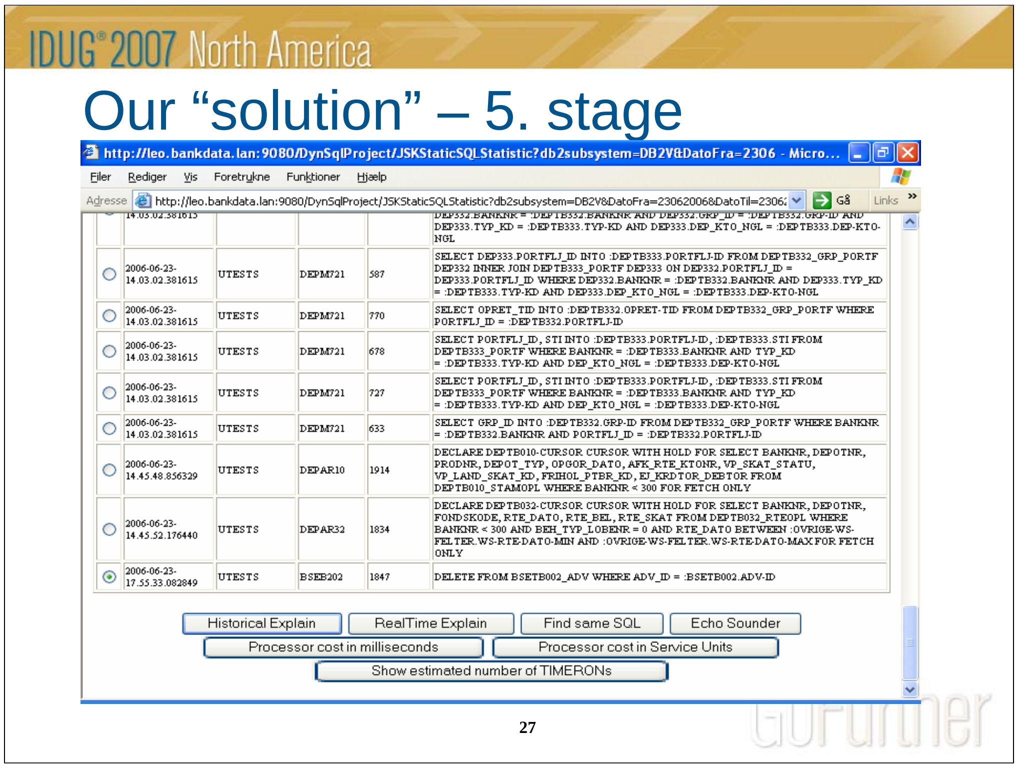
- [✓ No Extras]
- Show SQL with TS scans
- Show SQL with Non matching INDEX scans
- Show SQL with Matching INDEX scans
- Show SQL sorted on Timeron
- Show SQL sorted on Processor cost in milliseconds
- Show SQL sorted on Processor cost in Service Units
- Show SQL sorted on Echo Sounder

Show SQL for Package: ____

Just an example ! We could have included filtering on “processor cost” in milliseconds, service units and “estimated number of TIMERONS” as these were included as direct columns as well ! Here we added functionality to sort on these columns !!

26

On the first-version for static SQL we have a few more fields as we can both filter and sort.



IDUG® 2007 North America

Our “solution” – 5. stage

http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=2306 - Micro...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=23062006&DatoTil=2306
Gå
Links

	14.03.02.381615				DEP332.BANKNR = :DEPTB332.BANKNR AND DEP332.GRP_ID = :DEPTB332.GRP-ID AND DEP333.TYP_KD = :DEPTB333.TYP-KD AND DEP333.DEP_KTO_NGL = :DEPTB333.DEP-KTO-NGL
○	2006-06-23-14.03.02.381615	UTESTS	DEPM721	587	SELECT DEP333.PORTFLJ_ID INTO :DEPTB333.PORTFLJ-ID FROM DEPTB332_GRP_PORTF DEP332 INNER JOIN DEPTB333_PORTF DEP333 ON DEP332.PORTFLJ_ID = DEP333.PORTFLJ_ID WHERE DEP332.BANKNR = :DEPTB332.BANKNR AND DEP333.TYP_KD = :DEPTB333.TYP-KD AND DEP333.DEP_KTO_NGL = :DEPTB333.DEP-KTO-NGL
○	2006-06-23-14.03.02.381615	UTESTS	DEPM721	770	SELECT OPRET_TID INTO :DEPTB332.OPRET-TID FROM DEPTB332_GRP_PORTF WHERE PORTFLJ_ID = :DEPTB332.PORTFLJ-ID
○	2006-06-23-14.03.02.381615	UTESTS	DEPM721	678	SELECT PORTFLJ_ID, STI INTO :DEPTB333.PORTFLJ-ID, :DEPTB333.STI FROM DEPTB333_PORTF WHERE BANKNR = :DEPTB333.BANKNR AND TYP_KD = :DEPTB333.TYP-KD AND DEP_KTO_NGL = :DEPTB333.DEP-KTO-NGL
○	2006-06-23-14.03.02.381615	UTESTS	DEPM721	727	SELECT PORTFLJ_ID, STI INTO :DEPTB333.PORTFLJ-ID, :DEPTB333.STI FROM DEPTB333_PORTF WHERE BANKNR = :DEPTB333.BANKNR AND TYP_KD = :DEPTB333.TYP-KD AND DEP_KTO_NGL = :DEPTB333.DEP-KTO-NGL
○	2006-06-23-14.03.02.381615	UTESTS	DEPM721	633	SELECT GRP_ID INTO :DEPTB332.GRP-ID FROM DEPTB332_GRP_PORTF WHERE BANKNR = :DEPTB332.BANKNR AND PORTFLJ_ID = :DEPTB332.PORTFLJ-ID
○	2006-06-23-14.45.48.856329	UTESTS	DEPAR10	1914	DECLARE DEPTB010-CURSOR CURSOR WITH HOLD FOR SELECT BANKNR, DEPOTNR, PRODNR, DEPOT_TYP, OPGOR_DATO, AFK_RTE_KTONR, VP_SKAT_STATU, VP_LAND_SKAT_KD, FRIHOL_PTBR_KD, EJ_KRDTOR_DEBTOR FROM DEPTB010_STAMOPL WHERE BANKNR < 300 FOR FETCH ONLY
○	2006-06-23-14.45.52.176440	UTESTS	DEPAR32	1834	DECLARE DEPTB032-CURSOR CURSOR WITH HOLD FOR SELECT BANKNR, DEPOTNR, FONDSKODE, RTE_DATO, RTE_BEL, RTE_SKAT FROM DEPTB032_RTEOPL WHERE BANKNR < 300 AND BEH_TYP_LOBENR = 0 AND RTE_DATO BETWEEN :OVRIGE-WS-FELTER.WS-RTE-DATO-MIN AND :OVRIGE-WS-FELTER.WS-RTE-DATO-MAX FOR FETCH ONLY
◉	2006-06-23-17.55.33.082849	UTESTS	BSEB202	1847	DELETE FROM BSETB002_ADV WHERE ADV_ID = :BSETB002.ADV-ID

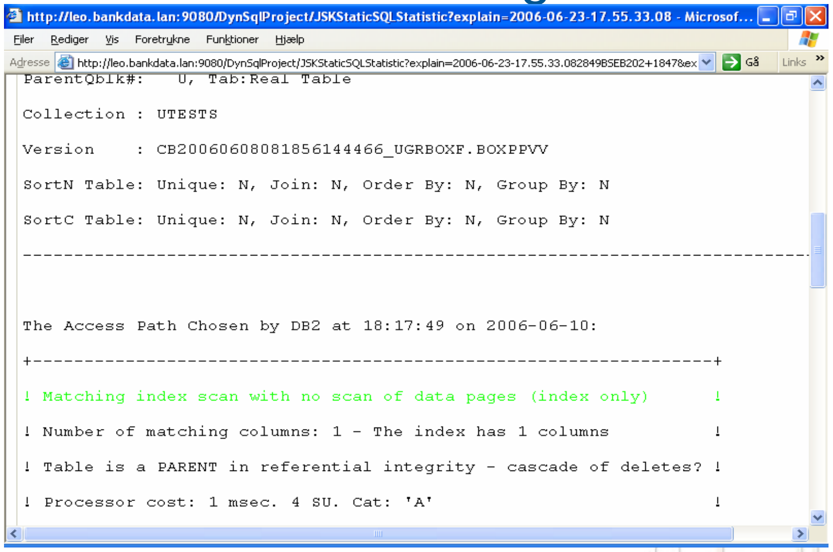
Historical Explain
RealTime Explain
Find same SQL
Echo Sounder
Processor cost in milliseconds
Processor cost in Service Units
Show estimated number of TIMERONs

27

GuruFinder

Here we have a report covering BINDs from one day. Please notice that we operate on statement number and package name and that the timestamp is the bind timestamp.

Our "solution" – 5. stage


http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-06-23-17.55.33.08 - Microsof...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-06-23-17.55.33.082849B5EB202+18478ex
Gå
Links

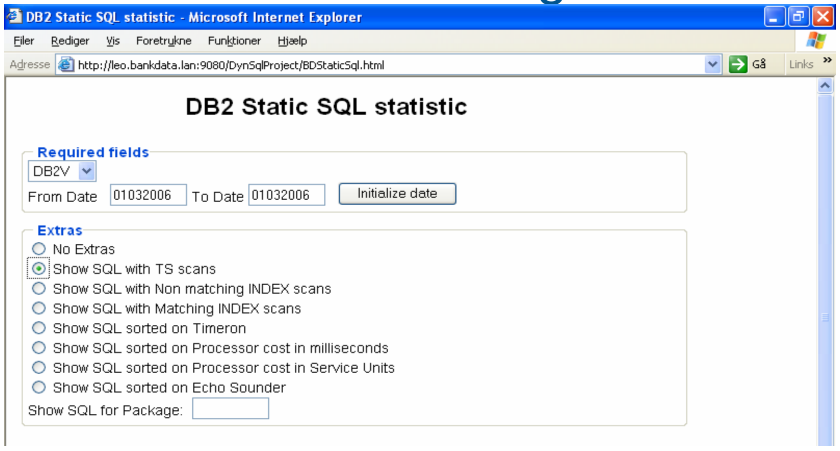
```
ParentQblk#:   0,  Tab:Real Table

Collection : UTESTS

Version    : CB20060608081856144466_UGRBOXF.BOXPPVV

SortN Table: Unique: N, Join: N, Order By: N, Group By: N

SortC Table: Unique: N, Join: N, Order By: N, Group By: N

-----------------------------------------------------------------------------


The Access Path Chosen by DB2 at 18:17:49 on 2006-06-10:

+-------------------------------------------------------------------+

! Matching index scan with no scan of data pages (index only)       !

! Number of matching columns: 1 - The index has 1 columns           !

! Table is a PARENT in referential integrity - cascade of deletes? !

! Processor cost: 1 msec. 4 SU. Cat: 'A'                            !
```

GoFurther

Like for Dynamic SQL we can display explain-report as produced the day of the bind.

IDUG® 2007 North America

Our "solution" – 5. stage


DB2 Static SQL statistic - Microsoft Internet Explorer

Adresse http://leo.bankdata.lan:9080/DynSqlProject/BDStaticSql.html

DB2 Static SQL statistic

Required fields

DB2V

From Date 01032006 To Date 01032006 [Initialize date]

Extras

- ○ No Extras
- ● Show SQL with TS scans
- ○ Show SQL with Non matching INDEX scans
- ○ Show SQL with Matching INDEX scans
- ○ Show SQL sorted on Timeron
- ○ Show SQL sorted on Processor cost in milliseconds
- ○ Show SQL sorted on Processor cost in Service Units
- ○ Show SQL sorted on Echo Sounder

Show SQL for Package:

Let's try to find packages that have statements with TS SCANS that was bound on March, 1. 2006 :

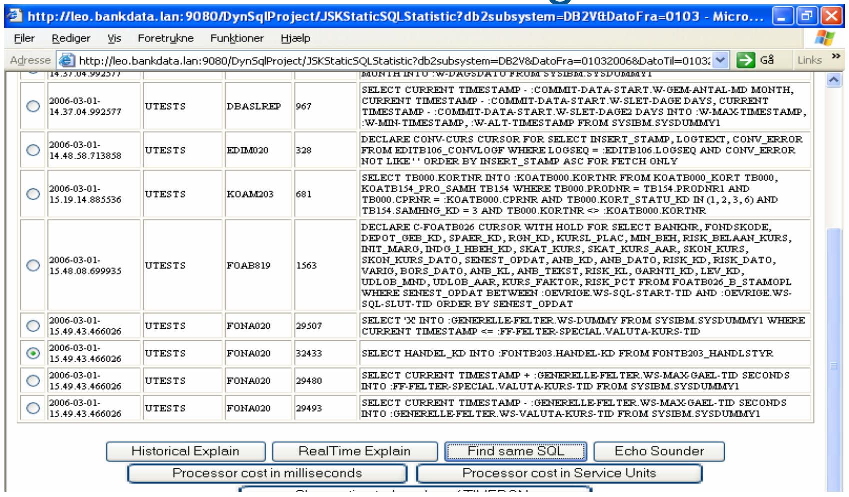
29

Gurutiner

Here we will show the power of filtering …… Lets take TableSpaceScans…..

IDUG® 2007 North America

Our “solution” – 5. stage



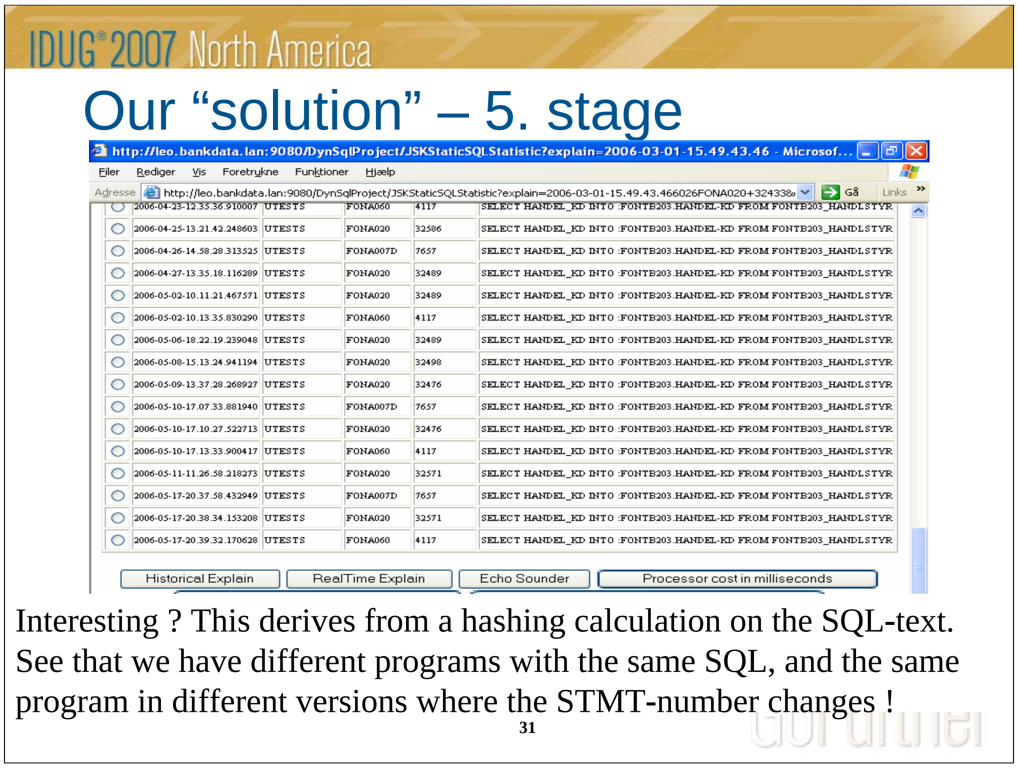
http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=0103 - Micro...

Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=01032006&DatoTil=0103:

	14.37.04.992577				MONTH INTO :W-DAGSDATO FROM SYSIBM.SYSDUMMY1
○	2006-03-01-14.37.04.992577	UTESTS	DBASLREP	967	SELECT CURRENT TIMESTAMP - :COMMIT-DATA-START.W-GEM-ANTAL-MD MONTH, CURRENT TIMESTAMP - :COMMIT-DATA-START.W-SLET-DAGE DAYS, CURRENT TIMESTAMP - :COMMIT-DATA-START.W-SLET-DAGE2 DAYS INTO :W-MAX-TIMESTAMP, :W-MIN-TIMESTAMP, :W-ALT-TIMESTAMP FROM SYSIBM.SYSDUMMY1
○	2006-03-01-14.48.58.713858	UTESTS	EDIM020	328	DECLARE CONV-CURS CURSOR FOR SELECT INSERT_STAMP, LOGTEXT, CONV_ERROR FROM EDITB106_CONVLOGF WHERE LOGSEQ = :EDITB106.LOGSEQ AND CONV_ERROR NOT LIKE ' ' ORDER BY INSERT_STAMP ASC FOR FETCH ONLY
○	2006-03-01-15.19.14.885536	UTESTS	KOAM203	681	SELECT TB000.KORTNR INTO :KOATB000.KORTNR FROM KOATB000_KORT TB000, KOATB154_PRO_SAMH TB154 WHERE TB000.PRODNR = TB154.PRODNR1 AND TB000.CPRNR = :KOATB000.CPRNR AND TB000.KORT_STATU_KD IN (1, 2, 3, 6) AND TB154.SAMHNG_KD = 3 AND TB000.KORTNR <> :KOATB000.KORTNR
○	2006-03-01-15.48.08.699935	UTESTS	FOAB819	1563	DECLARE C-FOATB026 CURSOR WITH HOLD FOR SELECT BANKNR, FONDSKODE, DEPOT_GEB_KD, SPAER_KD, RGN_KD, KURSL_PLAC, MIN_BEH, RISK_BELAAN_KURS, INIT_MARG, INDG_I_HBEH_KD, SKAT_KURS, SKAT_KURS_AAR, SKON_KURS, SKON_KURS_DATO, SENEST_OPDAT, ANB_KD, ANB_DATO, RISK_KD, RISK_DATO, VARIG, BORS_DATO, ANB_KL, ANB_TEKST, RISK_KL, GARNTI_KD, LEV_KD, UDLOB_MND, UDLOB_AAR, KURS_FAKTOR, RISK_PCT FROM FOATB026_B_STAMOPL WHERE SENEST_OPDAT BETWEEN :OEVRIGE.WS-SQL-START-TID AND :OEVRIGE.WS-SQL-SLUT-TID ORDER BY SENEST_OPDAT
○	2006-03-01-15.49.43.466026	UTESTS	FONA020	29507	SELECT 'X' INTO :GENERELLE-FELTER.WS-DUMMY FROM SYSIBM.SYSDUMMY1 WHERE CURRENT TIMESTAMP <= :FF-FELTER-SPECIAL.VALUTA-KURS-TID
●	2006-03-01-15.49.43.466026	UTESTS	FONA020	32433	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-03-01-15.49.43.466026	UTESTS	FONA020	29480	SELECT CURRENT TIMESTAMP + :GENERELLE-FELTER.WS-MAX-GAEL-TID SECONDS INTO :FF-FELTER-SPECIAL.VALUTA-KURS-TID FROM SYSIBM.SYSDUMMY1
○	2006-03-01-15.49.43.466026	UTESTS	FONA020	29493	SELECT CURRENT TIMESTAMP - :GENERELLE-FELTER.WS-MAX-GAEL-TID SECONDS INTO :GENERELLE-FELTER.WS-VALUTA-KURS-TID FROM SYSIBM.SYSDUMMY1

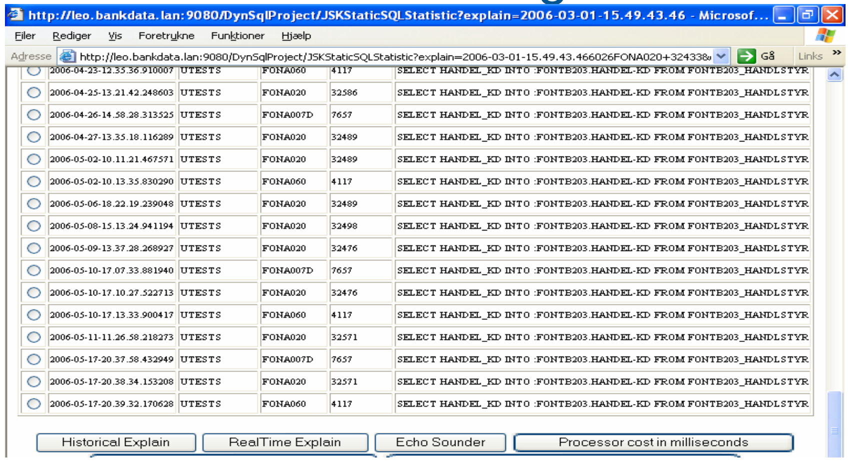
That was easy ! Responsetime is a fraction of a second as we have good indexes. Lets try to find anybody with the same SQL :

Here they are ..



IDUG® 2007 North America

Our “solution” – 5. stage


http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-03-01-15.49.43.46 - Microsof...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-03-01-15.49.43.466026FONA020+324338
Gå
Links

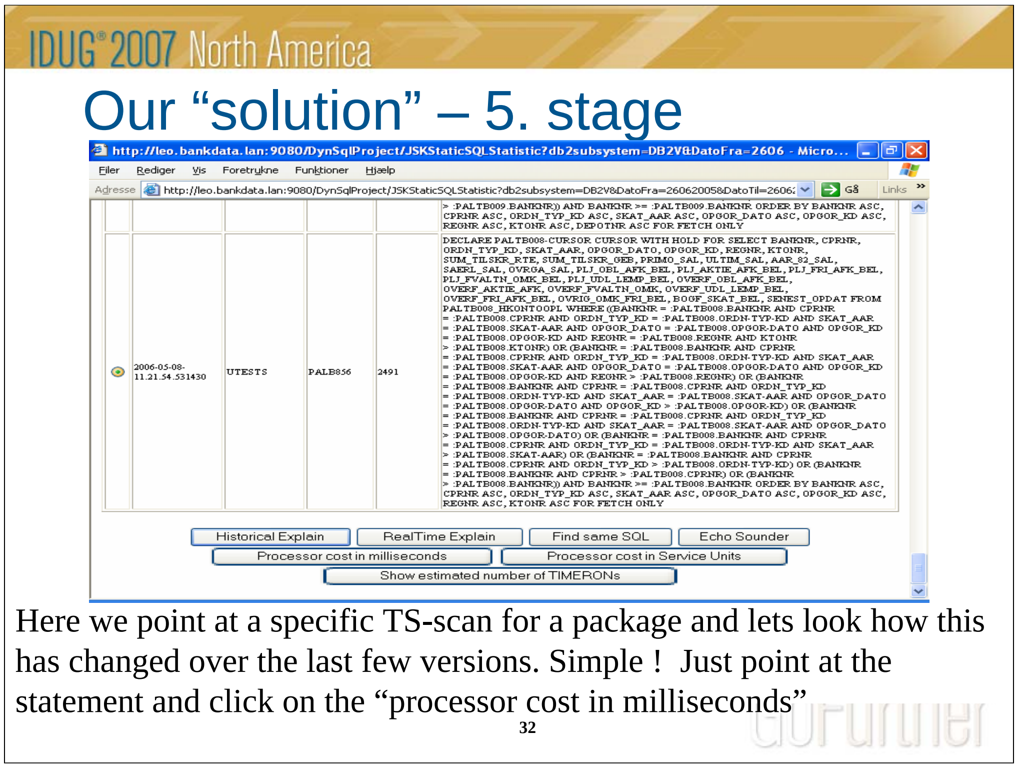
○	2006-04-23-12.35.36.910007	UTESTS	FONA060	4117	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-04-25-13.21.42.248603	UTESTS	FONA020	32586	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-04-26-14.58.28.313525	UTESTS	FONA007D	7657	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-04-27-13.35.18.116289	UTESTS	FONA020	32489	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-02-10.11.21.467571	UTESTS	FONA020	32489	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-02-10.13.35.830290	UTESTS	FONA060	4117	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-06-18.22.19.239048	UTESTS	FONA020	32489	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-08-15.13.24.941194	UTESTS	FONA020	32498	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-09-13.37.28.268927	UTESTS	FONA020	32476	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-10-17.07.33.881940	UTESTS	FONA007D	7657	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-10-17.10.27.522713	UTESTS	FONA020	32476	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-10-17.13.33.900417	UTESTS	FONA060	4117	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-11-11.26.58.218273	UTESTS	FONA020	32571	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-17-20.37.58.432949	UTESTS	FONA007D	7657	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-17-20.38.34.153208	UTESTS	FONA020	32571	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR
○	2006-05-17-20.39.32.170628	UTESTS	FONA060	4117	SELECT HANDEL_KD INTO :FONTB203.HANDEL-KD FROM FONTB203_HANDLSTYR

Interesting ? This derives from a hashing calculation on the SQL-text. See that we have different programs with the same SQL, and the same program in different versions where the STMT-number changes !

31

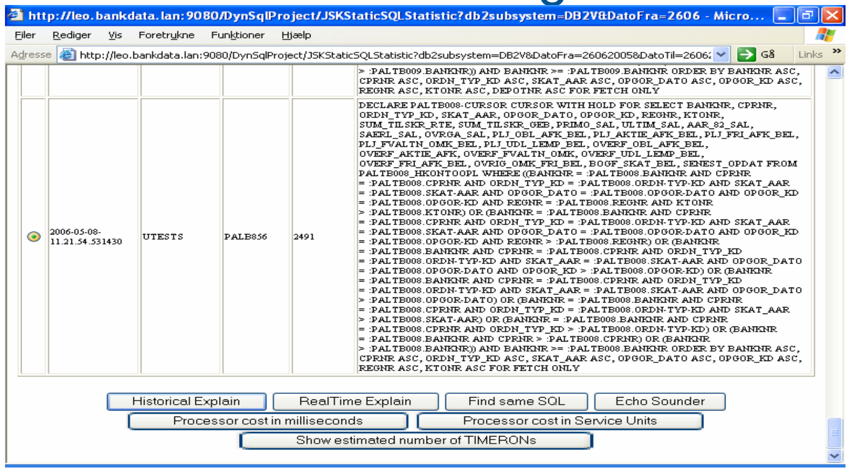
On this slide we have picked the “SHOW SAME SQL”-button. We take the hashing-key for the SQL and find anyone having exactly the same SQL. This can be beneficially in several aspects :

1) If the statement number changes in a COBOL-program it will still be found and can be compared
2) If you find one statement (dynamic or static) that should be improved by a rewrite, you can find all other SQL that should be changed as well !



IDUG® 2007 North America

Our “solution” – 5. stage



http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=2606 - Micro...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?db2subsystem=DB2V&DatoFra=26062005&DatoTil=2606
Gå
Links

					> :PALTB009.BANKNR)) AND BANKNR >= :PALTB009.BANKNR ORDER BY BANKNR ASC, CPRNR ASC, ORDN_TYP_KD ASC, SKAT_AAR ASC, OPGOR_DATO ASC, OPGOR_KD ASC, REGNR ASC, KTONR ASC, DEPOTNR ASC FOR FETCH ONLY
●	2006-05-08-11.21.54.531430	UTESTS	PALB856	2491	DECLARE PALTB008-CURSOR CURSOR WITH HOLD FOR SELECT BANKNR, CPRNR, ORDN_TYP_KD, SKAT_AAR, OPGOR_DATO, OPGOR_KD, REGNR, KTONR, SUM_TILSKR_RTE, SUM_TILSKR_GEB, PRIMO_SAL, ULTIM_SAL, AAR_82_SAL, SAERL_SAL, OVRGA_SAL, PLJ_OBL_AFK_BEL, PLJ_AKTIE_AFK_BEL, PLJ_FRI_AFK_BEL, PLJ_FVALTN_OMK_BEL, PLJ_UDL_LEMP_BEL, OVERF_OBL_AFK_BEL, OVERF_AKTIE_AFK, OVERF_FVALTN_OMK, OVERF_UDL_LEMP_BEL, OVERF_FRI_AFK_BEL, OVRIG_OMK_FRI_BEL, BOGF_SKAT_BEL, SENEST_OPDAT FROM PALTB008_HKONTOOPL WHERE ((BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD = :PALTB008.ORDN-TYP-KD AND SKAT_AAR = :PALTB008.SKAT-AAR AND OPGOR_DATO = :PALTB008.OPGOR-DATO AND OPGOR_KD = :PALTB008.OPGOR-KD AND REGNR = :PALTB008.REGNR AND KTONR > :PALTB008.KTONR) OR (BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD = :PALTB008.ORDN-TYP-KD AND SKAT_AAR = :PALTB008.SKAT-AAR AND OPGOR_DATO = :PALTB008.OPGOR-DATO AND OPGOR_KD = :PALTB008.OPGOR-KD AND REGNR > :PALTB008.REGNR) OR (BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD = :PALTB008.ORDN-TYP-KD AND SKAT_AAR = :PALTB008.SKAT-AAR AND OPGOR_DATO = :PALTB008.OPGOR-DATO AND OPGOR_KD > :PALTB008.OPGOR-KD) OR (BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD = :PALTB008.ORDN-TYP-KD AND SKAT_AAR = :PALTB008.SKAT-AAR AND OPGOR_DATO > :PALTB008.OPGOR-DATO) OR (BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD = :PALTB008.ORDN-TYP-KD AND SKAT_AAR > :PALTB008.SKAT-AAR) OR (BANKNR = :PALTB008.BANKNR AND CPRNR = :PALTB008.CPRNR AND ORDN_TYP_KD > :PALTB008.ORDN-TYP-KD) OR (BANKNR = :PALTB008.BANKNR AND CPRNR > :PALTB008.CPRNR) OR (BANKNR > :PALTB008.BANKNR)) AND BANKNR >= :PALTB008.BANKNR ORDER BY BANKNR ASC, CPRNR ASC, ORDN_TYP_KD ASC, SKAT_AAR ASC, OPGOR_DATO ASC, OPGOR_KD ASC, REGNR ASC, KTONR ASC FOR FETCH ONLY

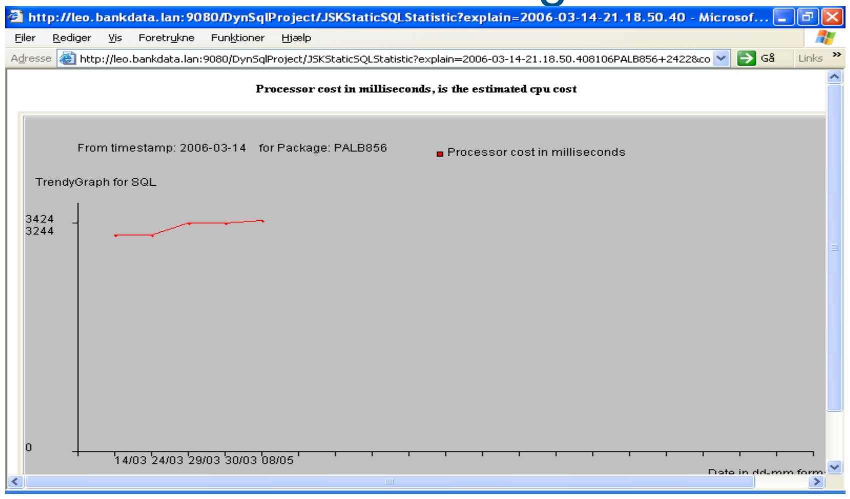
Historical Explain
RealTime Explain
Find same SQL
Echo Sounder
Processor cost in milliseconds
Processor cost in Service Units
Show estimated number of TIMERONs

Here we point at a specific TS-scan for a package and lets look how this has changed over the last few versions. Simple ! Just point at the statement and click on the “processor cost in milliseconds”

32

Here we have a specific statement in a package and we will show graphically how this has effected . ……..

Our “solution” – 5. stage

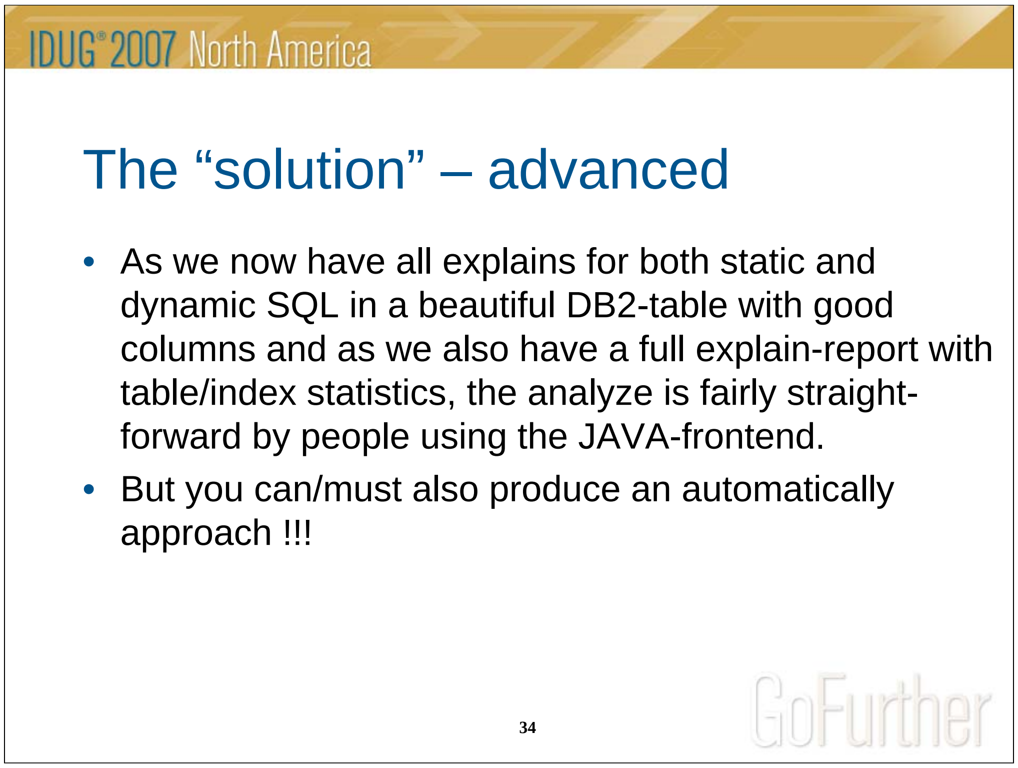

http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-03-14-21.18.50.40 - Microsof...

http://leo.bankdata.lan:9080/DynSqlProject/JSKStaticSQLStatistic?explain=2006-03-14-21.18.50.408106PALB856+2422&co

Processor cost in milliseconds, is the estimated cpu cost

And we get a beautiful, graphically view of the costs over the last 4 Binds. No matter if the statement-number has changed !!!

According to the optimizer some 10% more expensive….


IDUG® 2007 North America

The “solution” – advanced

- As we now have all explains for both static and dynamic SQL in a beautiful DB2-table with good columns and as we also have a full explain-report with table/index statistics, the analyze is fairly straight-forward by people using the JAVA-frontend.
- But you can/must also produce an automatically approach !!!

34

GoFurther

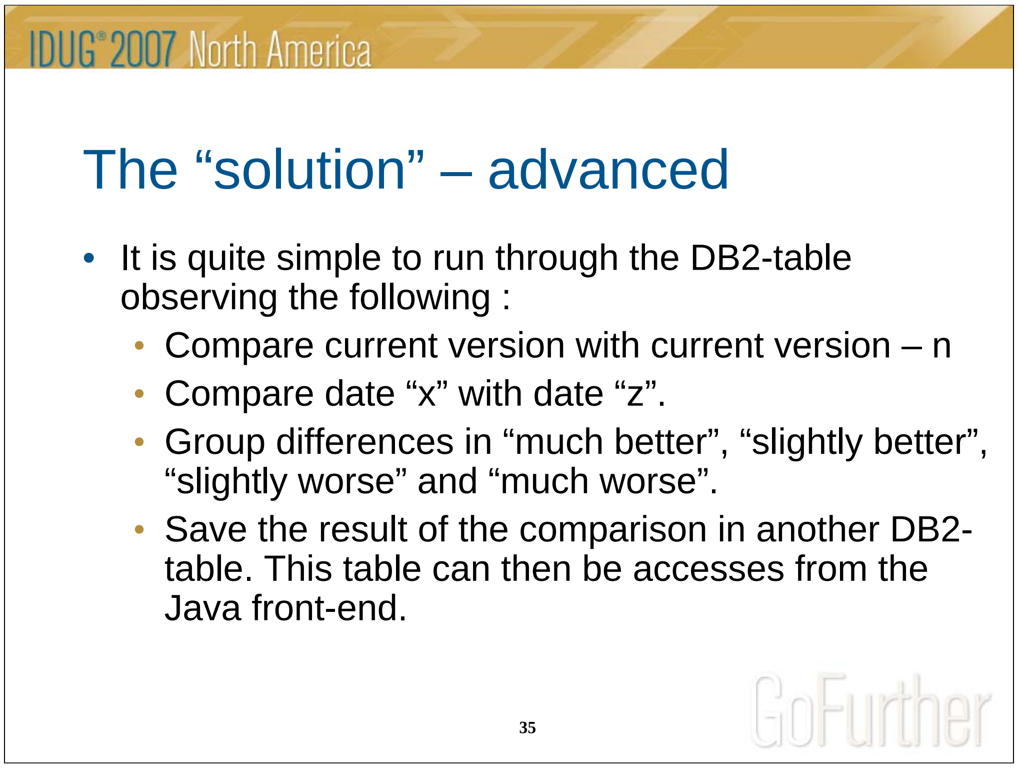
Now we can find everything manually. But certainly we need to automate things !!

-

We make some comparisons-run that will compare from one bind to the other, looking at either versions or dates. The difference in the optimizer figures are then evaluated as “the same”,”better”,”much better”,”worse” or “much worse”. The result of this comparison are stored in yet another table, and all we need is a alarming mechanism to mail DBA’s if differences of a given magnitude is seen.

-

A new front-end entrance is build. In java so it can be accessed from our intranet.


IDUG® 2007 North America

The “solution” – advanced

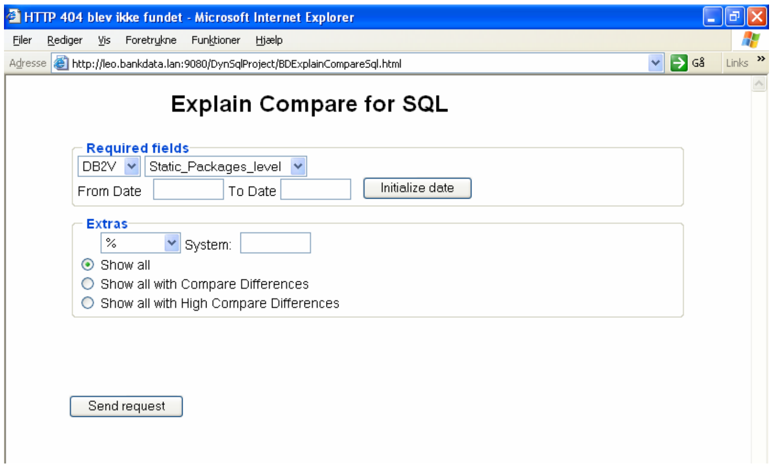
- It is quite simple to run through the DB2-table observing the following :
  - Compare current version with current version – n
  - Compare date “x” with date “z”.
  - Group differences in “much better”, “slightly better”, “slightly worse” and “much worse”.
  - Save the result of the comparison in another DB2-table. This table can then be accesses from the Java front-end.

35

GoFurther

Se previous foil …

The “solution” – advanced


HTTP 404 blev ikke fundet - Microsoft Internet Explorer
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/BDExplainCompareSql.html
Gå Links
Explain Compare for SQL
Required fields
DB2V
Static_Packages_level
From Date
To Date
Initialize date
Extras
%
System:
Show all
Show all with Compare Differences
Show all with High Compare Differences
Send request

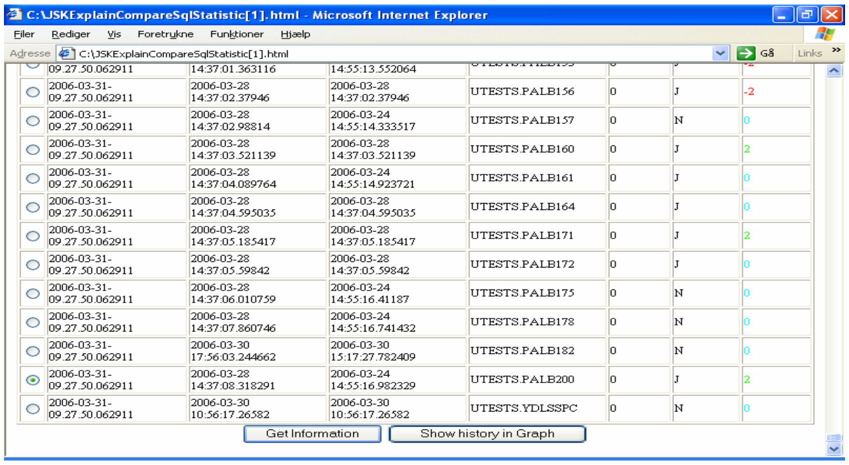
The entrance panel on our intra-net. We can filter on static/dynamic SQL, level of differences, date-span and program-names.

36

We made a very simple panel (first version), where we filter on packagename, and magnitude of differences.

IDUG® 2007 North America

The “solution” – advanced


C:\JSKExplainCompareSqlStatistic[1].html - Microsoft Internet Explorer

Filer Rediger Vis Foretrukne Funktioner Hjælp

Adresse C:\JSKExplainCompareSqlStatistic[1].html

○	09.27.50.062911	14:37:01.363116	14:55:13.552064	[illegible]	[illegible]	[illegible]	[illegible]
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:02.37946	2006-03-28 14:37:02.37946	UTESTS.PALB156	0	J	-2
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:02.98814	2006-03-24 14:55:14.333517	UTESTS.PALB157	0	N	0
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:03.521139	2006-03-28 14:37:03.521139	UTESTS.PALB160	0	J	2
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:04.089764	2006-03-24 14:55:14.923721	UTESTS.PALB161	0	J	0
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:04.595035	2006-03-28 14:37:04.595035	UTESTS.PALB164	0	J	0
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:05.185417	2006-03-28 14:37:05.185417	UTESTS.PALB171	0	J	2
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:05.59842	2006-03-28 14:37:05.59842	UTESTS.PALB172	0	J	0
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:06.010759	2006-03-24 14:55:16.41187	UTESTS.PALB175	0	N	0
○	2006-03-31-09.27.50.062911	2006-03-28 14:37:07.860746	2006-03-24 14:55:16.741432	UTESTS.PALB178	0	N	0
○	2006-03-31-09.27.50.062911	2006-03-30 17:56:03.244662	2006-03-30 15:17:27.782409	UTESTS.PALB182	0	N	0
◉	2006-03-31-09.27.50.062911	2006-03-28 14:37:08.318291	2006-03-24 14:55:16.982329	UTESTS.PALB200	0	J	2
○	2006-03-31-09.27.50.062911	2006-03-30 10:56:17.26582	2006-03-30 10:56:17.26582	UTESTS.YDLSSPC	0	N	0

Here we see one line per program with an indication of how serious the differences are. Lets look at PALB200 :

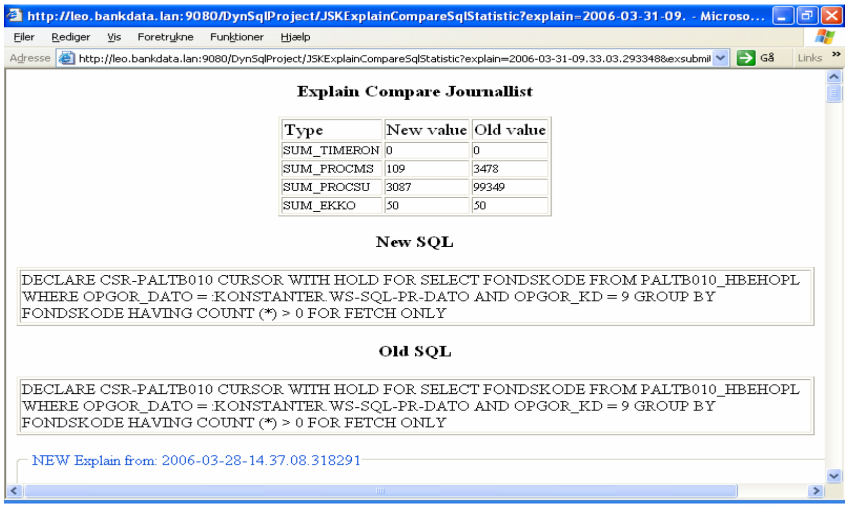
37

GoFurther

Here all result of a comparison is shown. Please notice the last-column that describes the differences in cost. It will be -2 for much worse, -1 for slightly worse, 0 for the same, +1 for slightly better and +2 for much better.

IDUG® 2007 North America

The “solution” – advanced


http://leo.bankdata.lan:9080/DynSqlProject/JSKExplainCompareSqlStatistic?explain=2006-03-31-09. - Microso...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKExplainCompareSqlStatistic?explain=2006-03-31-09.33.03.293348&exsubmi
Gå
Links

Explain Compare Journallist

Type	New value	Old value
SUM_TIMERON	0	0
SUM_PROCMS	109	3478
SUM_PROCSU	3087	99349
SUM_EKKO	50	50

New SQL

```
DECLARE CSR-PALTB010 CURSOR WITH HOLD FOR SELECT FONDSKODE FROM PALTB010_HBEHOPL
WHERE OPGOR_DATO = :KONSTANTER.WS-SQL-PR-DATO AND OPGOR_KD = 9 GROUP BY
FONDSKODE HAVING COUNT (*) > 0 FOR FETCH ONLY
```

Old SQL

```
DECLARE CSR-PALTB010 CURSOR WITH HOLD FOR SELECT FONDSKODE FROM PALTB010_HBEHOPL
WHERE OPGOR_DATO = :KONSTANTER.WS-SQL-PR-DATO AND OPGOR_KD = 9 GROUP BY
FONDSKODE HAVING COUNT (*) > 0 FOR FETCH ONLY
```

NEW Explain from: 2006-03-28-14.37.08.318291

We show a explanation of the difference and an explain report.

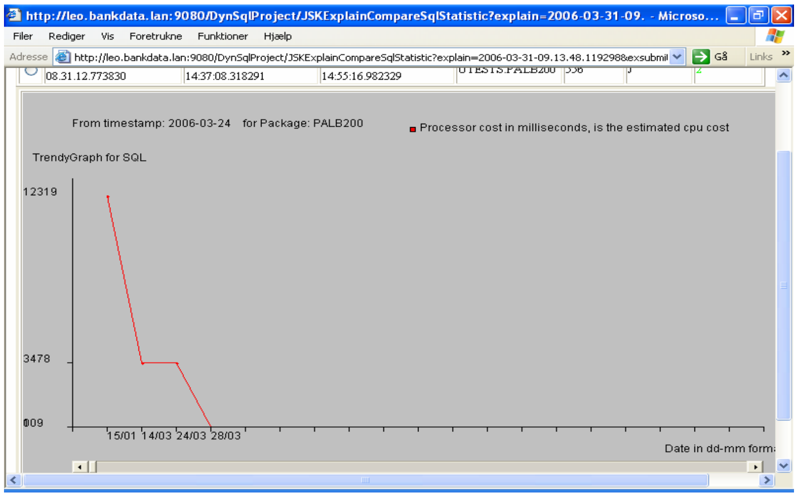
38

GoFurther

For every difference we can show all relevant figures including the full explain-report, both for ’before’ and ‘after’.

IDUG® 2007 North America

The “solution” – advanced


http://leo.bankdata.lan:9080/DynSqlProject/JSKExplainCompareSqlStatistic?explain=2006-03-31-09. - Microso...
Filer Rediger Vis Foretrukne Funktioner Hjælp
Adresse http://leo.bankdata.lan:9080/DynSqlProject/JSKExplainCompareSqlStatistic?explain=2006-03-31-09.13.48.119298&exsubmit
Gå
Links
08.31.12.773830
14:37:08.318291
14:55:16.982329
UTESTS.PALB200

From timestamp: 2006-03-24 for Package: PALB200

🟥 Processor cost in milliseconds, is the estimated cpu cost

TrendyGraph for SQL

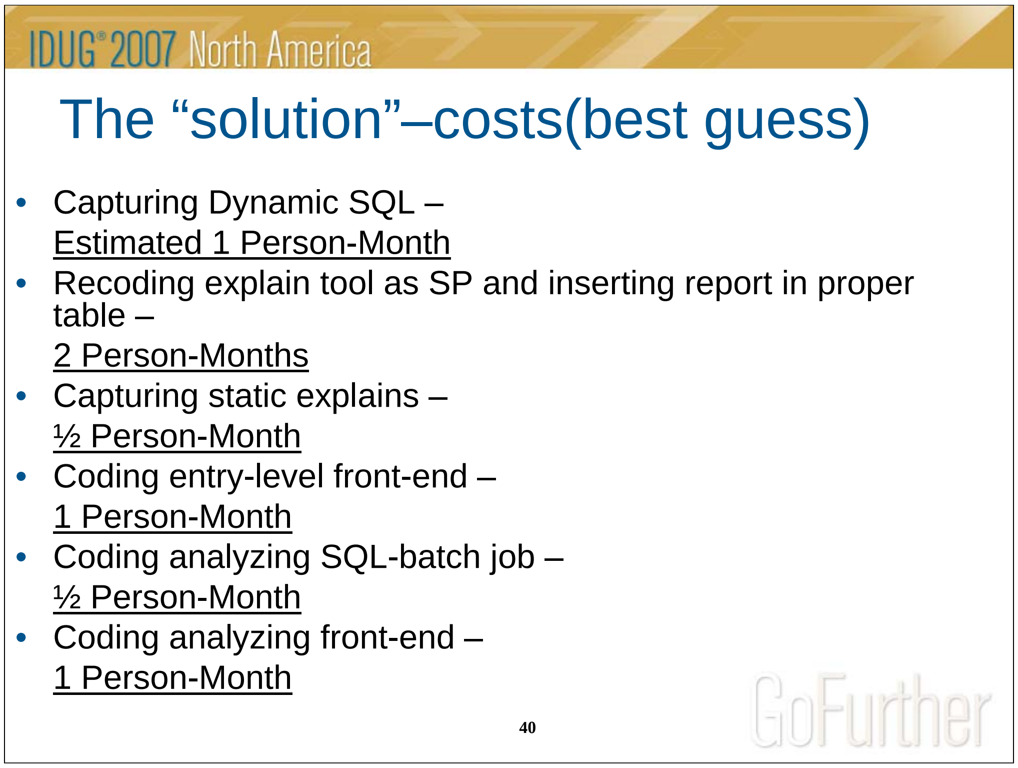
12319
3478
009
15/01
14/03
24/03
28/03
Date in dd-mm form

For a specific statement we can even graph the processor cost.

39

GoFurther

And the difference from the optimizer figures can be shown graphically.


IDUG® 2007 North America

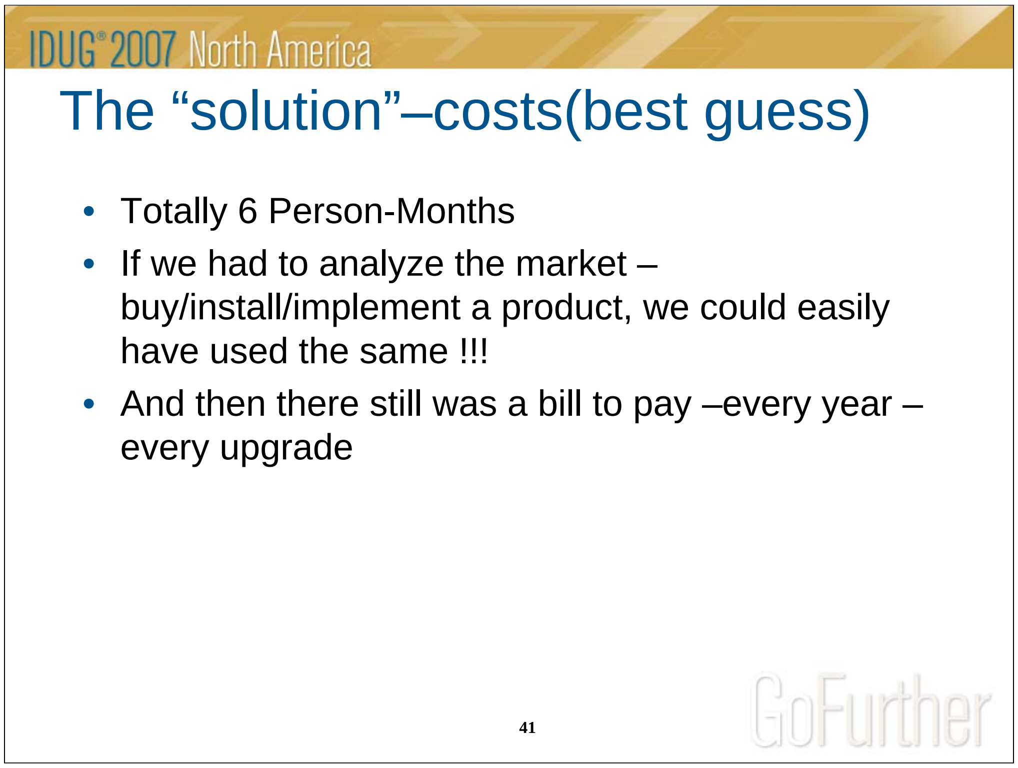
The “solution”–costs(best guess)

- Capturing Dynamic SQL –
  Estimated 1 Person-Month
- Recoding explain tool as SP and inserting report in proper table –
  2 Person-Months
- Capturing static explains –
  ½ Person-Month
- Coding entry-level front-end –
  1 Person-Month
- Coding analyzing SQL-batch job –
  ½ Person-Month
- Coding analyzing front-end –
  1 Person-Month

40

GoFurther

If things like this can be sliced into different phases, where each one will produce benefit, it will not cost that much. And you will be encouraged to continue….


IDUG® 2007 North America

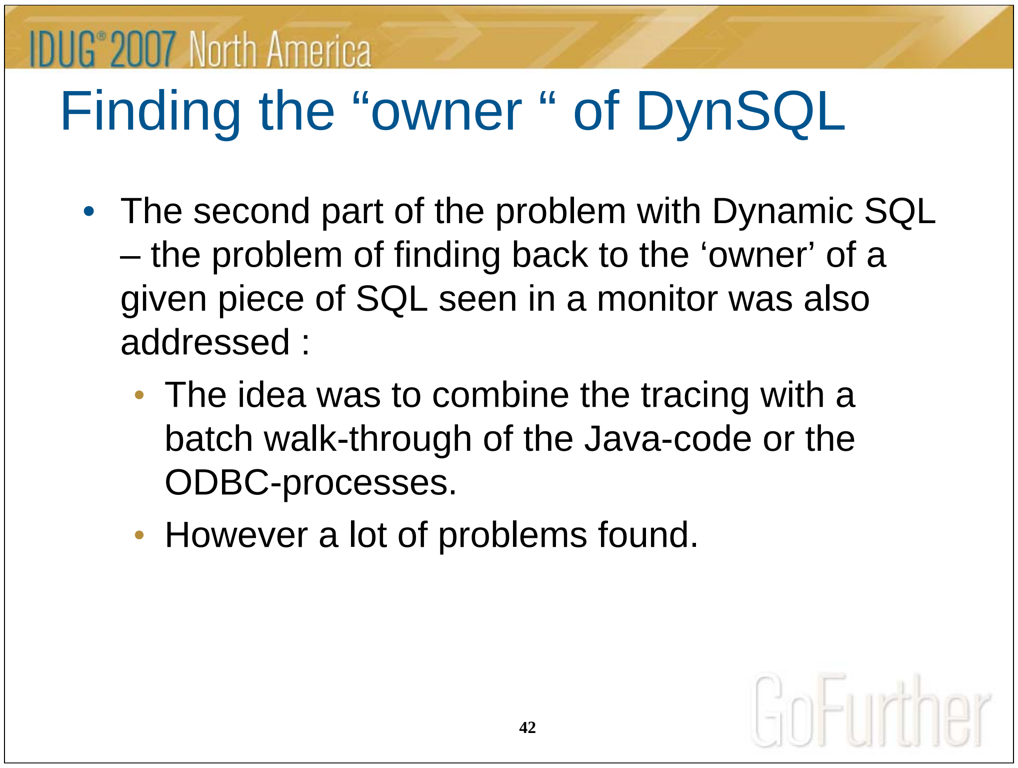
The “solution”–costs(best guess)

- Totally 6 Person-Months
- If we had to analyze the market – buy/install/implement a product, we could easily have used the same !!!
- And then there still was a bill to pay –every year – every upgrade

41

GoFurther

A very good experience !!



IDUG® 2007 North America

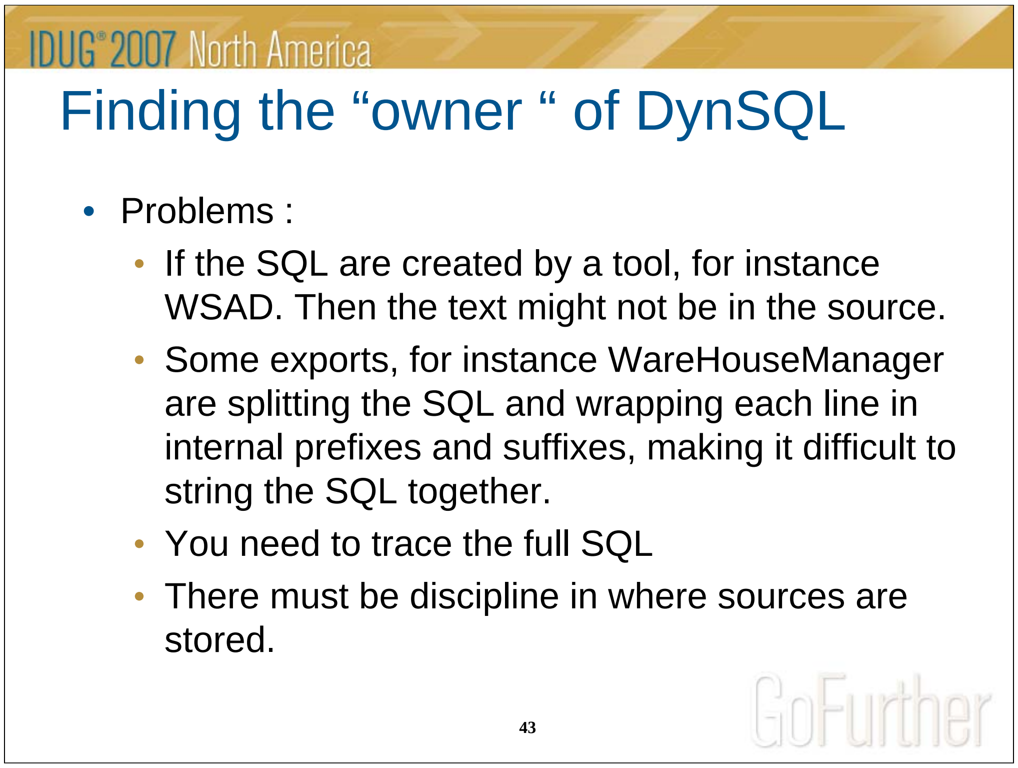
Finding the “owner “ of DynSQL

- The second part of the problem with Dynamic SQL – the problem of finding back to the ‘owner’ of a given piece of SQL seen in a monitor was also addressed :
  - The idea was to combine the tracing with a batch walk-through of the Java-code or the ODBC-processes.
  - However a lot of problems found.

42

GoFurther

The ‘exporting’ of the source from different sources like JAVA and DataWareHousing repository was an exercise that sounded simple, but turned out as rather complicated.


IDUG® 2007 North America

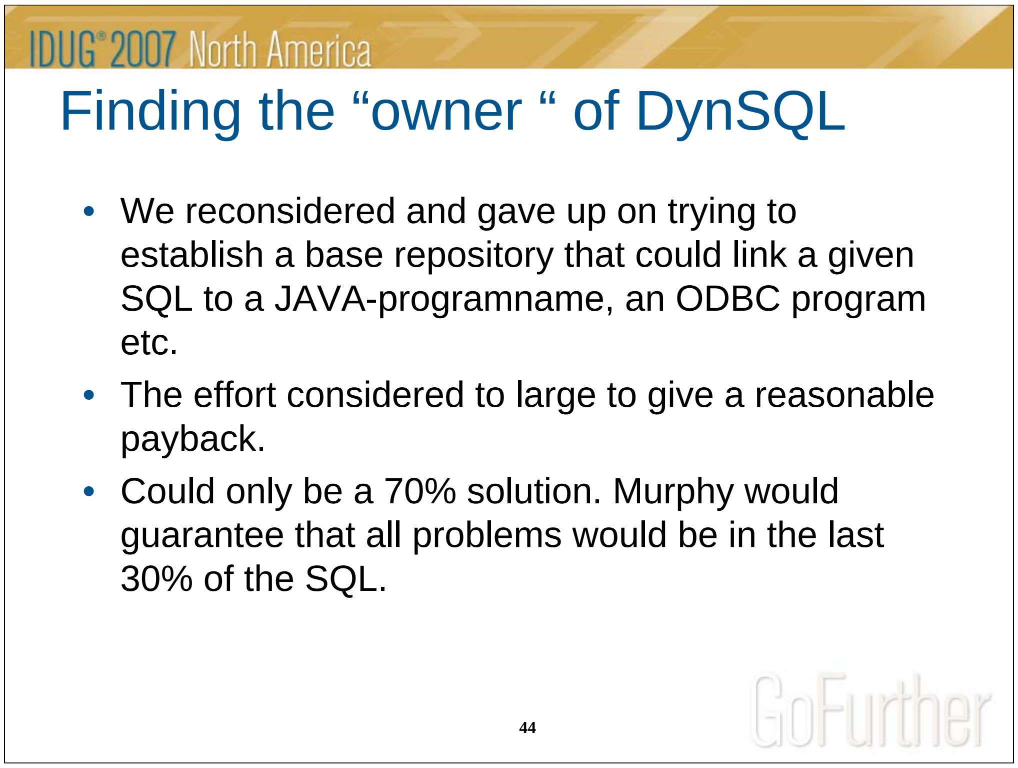
Finding the “owner “ of DynSQL

- Problems :
  - If the SQL are created by a tool, for instance WSAD. Then the text might not be in the source.
  - Some exports, for instance WareHouseManager are splitting the SQL and wrapping each line in internal prefixes and suffixes, making it difficult to string the SQL together.
  - You need to trace the full SQL
  - There must be discipline in where sources are stored.

43

GoFurther

There will be a great deal of coding in some instances as the SQL is not that handy from these sources. And in some times impossible. As we still have not found the time for recoding in V8 to trace the full SQL-text, it was even more complicated as many of these Dynamic SQL turned out to be extremely long.



IDUG® 2007 North America

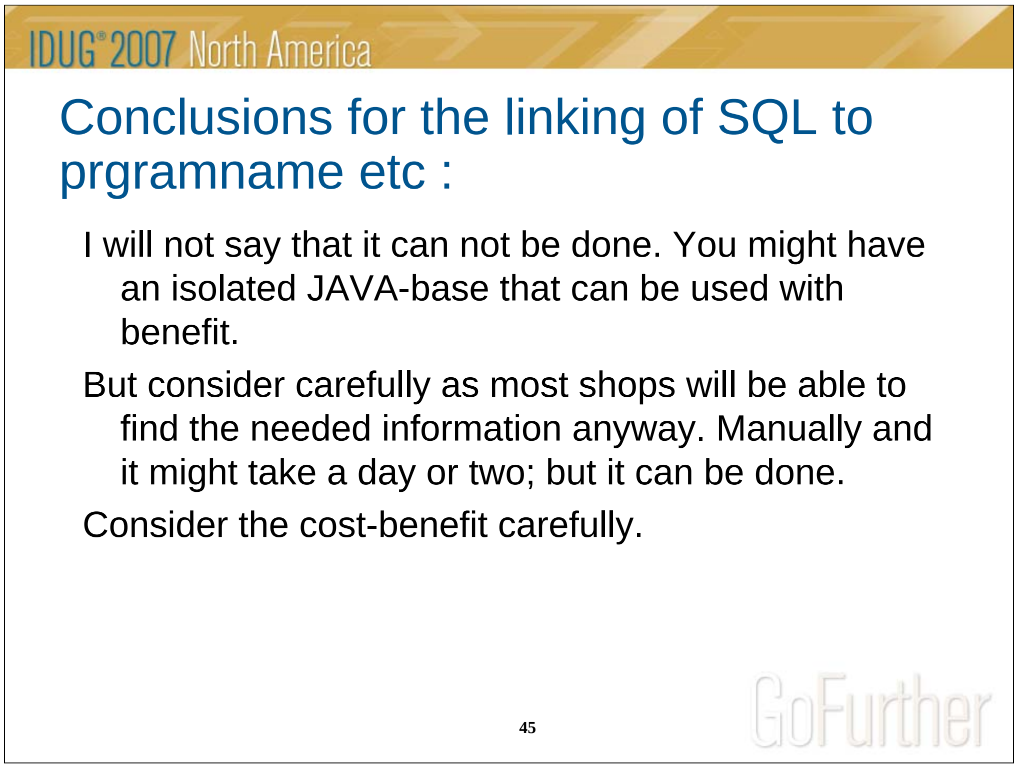
Finding the "owner " of DynSQL

- We reconsidered and gave up on trying to establish a base repository that could link a given SQL to a JAVA-programname, an ODBC program etc.
- The effort considered to large to give a reasonable payback.
- Could only be a 70% solution. Murphy would guarantee that all problems would be in the last 30% of the SQL.

44

GoFurther

We might give it another try in the future, but warehousing was our primary target, and there seem to be re-considerations concerning the running platform in our installation…



IDUG® 2007 North America

Conclusions for the linking of SQL to prgramname etc :

I will not say that it can not be done. You might have an isolated JAVA-base that can be used with benefit.

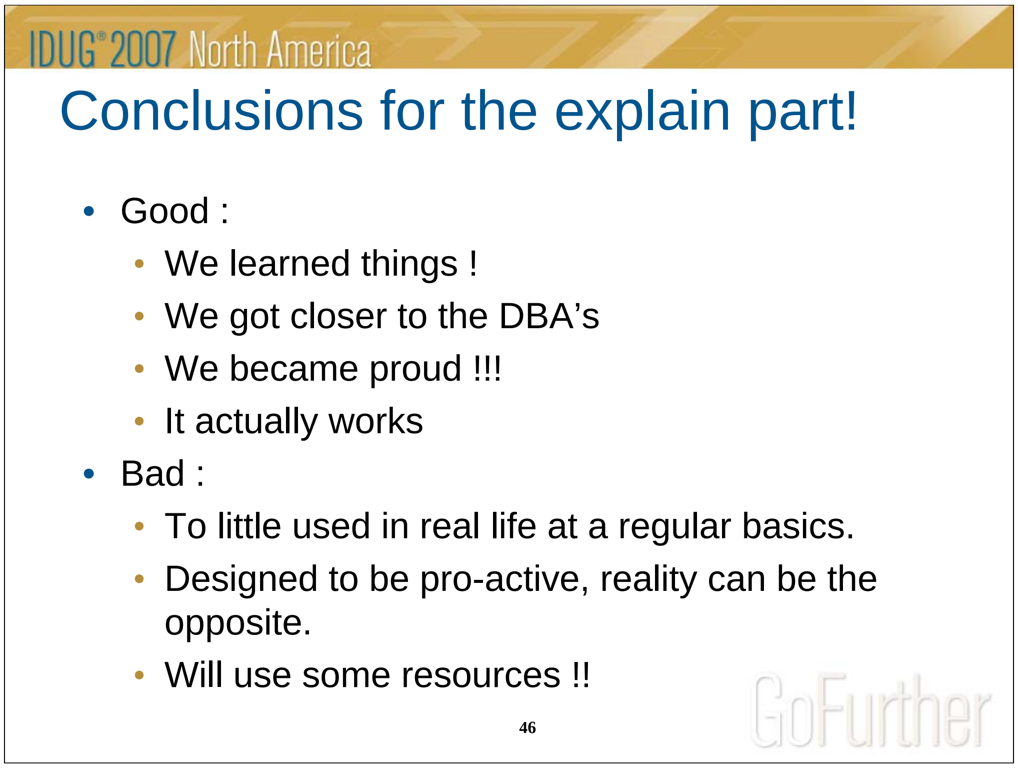
But consider carefully as most shops will be able to find the needed information anyway. Manually and it might take a day or two; but it can be done.

Consider the cost-benefit carefully.

45

GoFurther

So be careful…

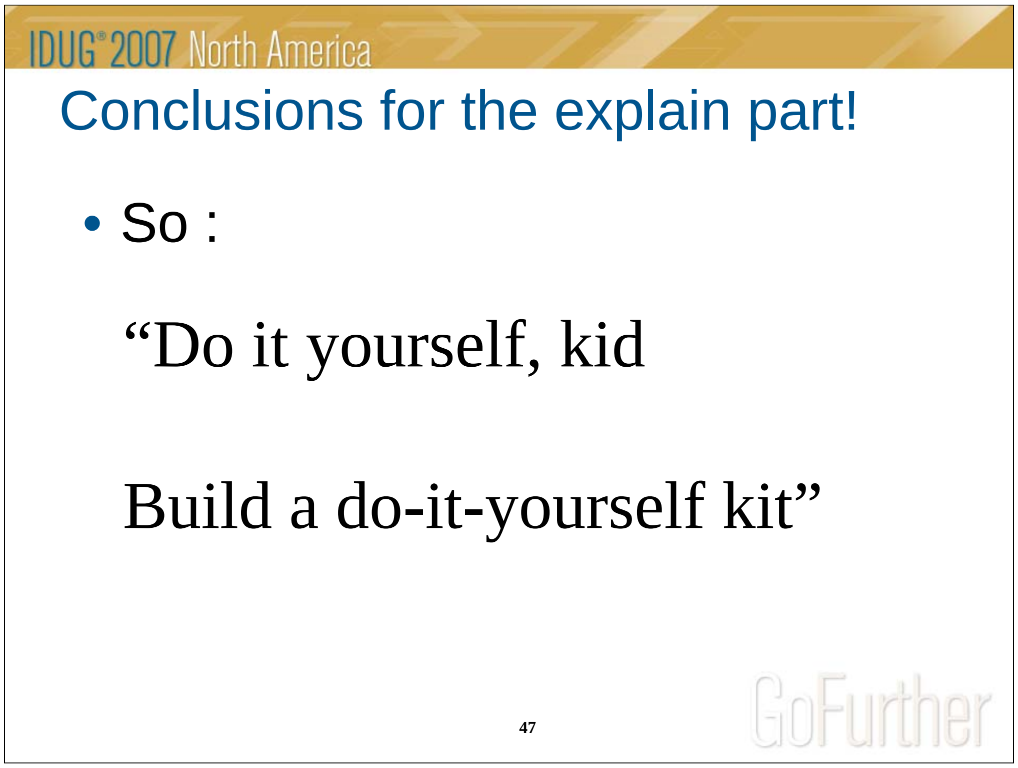

IDUG® 2007 North America

Conclusions for the explain part!

- Good :
  - We learned things !
  - We got closer to the DBA's
  - We became proud !!!
  - It actually works
- Bad :
  - To little used in real life at a regular basics.
  - Designed to be pro-active, reality can be the opposite.
  - Will use some resources !!

46

GoFurther



Conclusions for the explain part!

- So :

“Do it yourself, kid

Build a do-it-yourself kit”

GoFurther

IDUG® 2007 North America

Session: B09

SQL and Access Paths
– "A DIY way to survive"
– for both static and dynamic SQL

Frank Petersen

Bankdata, Denmark

fap@bankdata.dk

48

GoFurther