ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 9 - NUMMER 1, MEI 2014

OPEN CURSOR

DB2 viert dit jaar z'n 30ste verjaardag. Een product dus dat al veel evoluties heeft meegemaakt. En zich geleidelijk heeft aangepast aan een gewijzigde informatica-realiteit, zonder z'n eigenheid te verliezen.

Getuige o.a. de stap in de jaren 90, via OS2, naar LUW, de "gedistribueerde platformen" dus. Met weliswaar een totaal verschillende "code base" dan die voor z/OS, waardoor een doorwinterde z/OS DB2er soms z'n weg niet dadelijk vindt op LUW.

Gelukkig is er ondertussen veel veranderd. Windows desktops bieden een handige grafische front-end, ook voor database-onderhoud en software-ontwikkeling op mainframe. Anderzijds bieden recente ontwikkelingen zoals zLinux, BLU acceleration, PureData, ... steeds nieuwe manieren om de voordelen van de verschillende platformen te combineren.

IN DIT NUMMER:

Twee bijdragen in de context van de "multi-platform" realiteit waarin DB2 zich vandaag bevindt:

- "*Rational Developer for System z en DB2*" geeft concrete tips hoe u RDz efficiënt kunt inzetten bij het ontwikkelen van software die gebruik maakt van DB2. Tegelijkertijd vindt de DBA hier ook enkele interessante suggesties voor o.a. het visualiseren van de DB2-catalog. Een interessante grafische front-end dus voor alle gebruikers van DB2 voor z/OS!
- "*DB2 LUW V10 hidden treasures - administratieve views en routines*" geeft een gedetailleerd overzicht van hoe u, als DBA, de verschillende bronnen van DB2-metadata op LUW kunt in het oog houden. U vindt er suggesties voor het efficiënt inrichten van een "DB2 dashboard" zodat u potentiële problemen snel en gericht kunt opsporen en verhelpen.

1

CLOSE CURSOR

Zoals u ondertussen van ons gewoon bent, zullen we u ook in de volgende nummers up-to-date houden met technische (en minder technische) details over DB2, in het bijzonder natuurlijk de nieuwe mogelijkheden van DB2 11 voor z/OS, evenals de integratie van DB2 met gerelateerde technologie.

Rational Developer for System z en DB2

Gie Indesteege (ABIS)

Voor het ontwikkelen van toepassingen in COBOL of PL/1 vergt het inzetten van Rational Developer for System z (**RDz**) in plaats van de traditionele ISPF-omgeving een hele aanpassing voor de ontwikkelaar. Maar wist u dat ook het voorbereiden en testen van SQL-queries voor de toepassingen kan gebeuren met behulp van deze RDz?

Deze bijdrage probeert een kort overzicht te geven van de mogelijkheden van het product voor de DB2-ontwikkelaar.

Achtergrond

Rational Developer for System z is ontstaan vanuit de **VisualAge** familie van producten. Nadat de broncode (herschreven in Java) werd overgedragen aan het Eclipse framework, werd de verdere ontwikkeling binnen IBM gebundeld in de **WebSphere Studio** producten, om uiteindelijk onder de Rational koepel te verzeilen.

Een stukje geschiedenis is te vinden op http://en.wikipedia.org/wiki/IBM_VisualAge

Rational Developer for System z is vandaag vooral gekend voor zijn flexibele mogelijkheden voor traditionele COBOL en PL/1 (batch) ontwikkeling, met daarnaast ook de nodige aandacht en ondersteuning voor online CICS TS en IMS/TM applicaties. De (Eclipse gebaseerde) IDE voorziet tal van mogelijkheden voor ontwikkelingsondersteuning aan de hand van bijvoorbeeld *code completion*, *code visualisering* en *syntax controle*, om er maar enkele te noemen.

Daarnaast is het (automatisch) genereren van de load module via geintegreerde JCL procedures voorzien, en voor de batch programma's kan een test JCL gegenereerd worden. On-line transacties kunnen dan weer (locaal) gedebugd worden via een integratie met de Debug tool voor z/OS.

Maar hoe zit het met de integratie met SQL en DB2?

Gebruik van database objecten en SQL in RDz

Reeds in de WebSphere Studio producten heeft men ondersteuning geboden voor de integratie van SQL. In het zogenaamde "*Data perspective*" was het mogelijk om een connectie op te zetten naar een database, en vervolgens DDL te genereren, stored procedures en user-gedefinieerde functies te definiëren. Ook een basis SQL editor was al voorhanden.

In de huidige **versie 9** van Rational Developer for System z, heeft men deze ondersteuning uitgebreid en gebaseerd op de mogelijkheden van **IBM Data Studio** (vervanger van DB2 Control Center). Dit

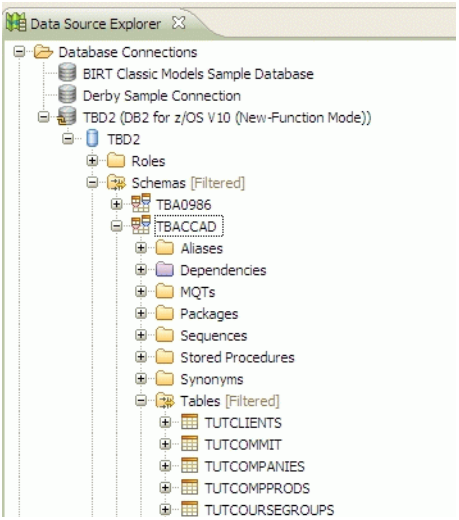
maakt het mogelijk voor zowel de DB2 DBA als de DB2 ontwikkelaar om allerhande DB2 activiteiten te ontplooien, inclusief de visualisatie van o.a. database objecten en structuren.

We kunnen in RDz verschillende perspectieven onderscheiden:

- **Data perspectief**: in de Data Source Explorer kunnen DBAs en ontwikkelaars database-connecties en -definities beheren. De connectiviteit wordt gerealiseerd met behulp van JDBC drivers, op basis van de definitie van connectieproperties.

Eens de databaseconnectie gerealiseerd, kunnen allerhande definities van databases, schema's, tabellen, stored procedures, SQL usergedefinieerde functies, en views, m.a.w. DB2 catalog informatie, geimporteerd worden.

Een voorbeeld van de *Data source explorer* en een connectie naar DB2:



Welke objecten getoond worden, hangt af van de filterdefinities voor schema, tabel en/of view. En natuurlijk speelt ook de beveiliging een rol.

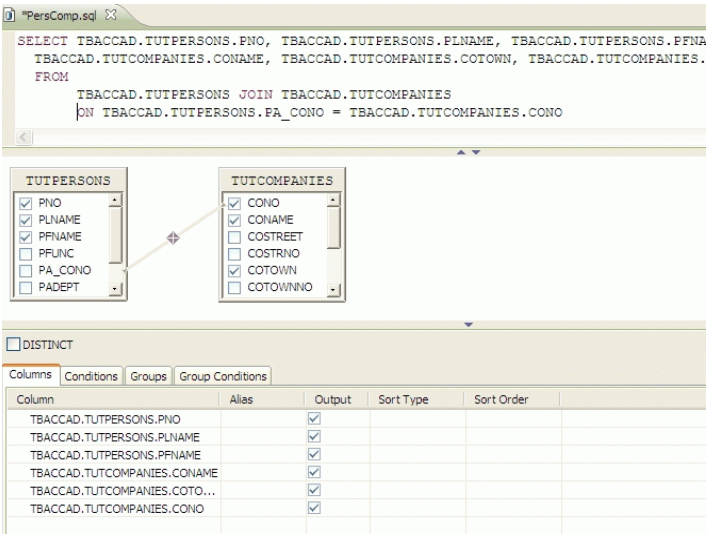
Vertrekkende van deze objecten kunnen dan allerhande acties ondernomen worden. Bijvoorbeeld kunnen één of meerdere tabellen geselecteerd worden waarvoor men

- DDL wil laten genereren, of
- voorbeelddata wilt tonen.

Elk object heeft natuurlijk zijn eigen mogelijke opties die via een pop-up menu (rechter muisklik) getoond worden.

Met behulp van de **SQL editor** (of builder) in de *Data Project Explorer* kunnen dan weer SQL statements worden gecreëerd en aangepast, en dit met de nodige ondersteuning van allerlei (grafische) wizards.

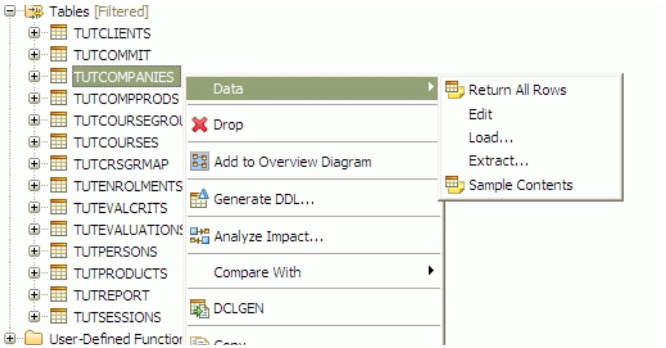
Een voorbeeld van de SQL editor:



Deze query werd automatisch samengesteld op basis van visuele manipulaties: de tabellen TUTPERSONS en TUTCOMPANIES werden vanuit de Data Source Explorer view in de SQL editor gedropt, en vervolgens werd de relatie tussen beide tabellen (op basis van het bedrijfsnummer, CONO=PA_CONO) grafisch aangeduid.

Geschreven (of gegenereerde) queries kunnen dan van hieruit rechtstreeks uitgevoerd en getest worden.

- In het **Database Development perspectief** komen dezelfde mogelijkheden terug, maar zijn er bijkomende opties voorzien voor bijvoorbeeld
  - het klaarmaken van testgegevens, of
  - het genereren van een DCLGEN copybook of include file.



Deze laatste optie laat straks toe om z/OS applicaties te baseren op deze copybook of include.

- Tenslotte is er nog het **Database Debug perspectief** voor het testen en stapsgewijs debuggen van applicaties met embedded SQL. Wordt in combinatie met het Debug Tool for z/OS gebruikt.

Ontwikkelen van applicaties met embedded SQL

Al deze perspectieven helpen de DB2 applicatie-ontwikkelaar om het SQL-onderdeel van een applicatie voor te bereiden en uit te testen.

- Testtabellen en testdata kunnen gemakkelijk aangemaakt worden,
- SQL-queries opbouwen wordt een stuk eenvoudiger met behulp van de query builder,
- de queries kunnen vervolgens uitgevoerd worden,

en dat alles zonder te verspringen van het ene venster naar het andere, zoals we in een ISPF-omgeving gewend zijn. De verschillende zichten (views) van de database objecten en acties staan broederlijk langs elkaar.

Van hieruit is het dan een kleine moeite om de gebruikte SQL over te dragen als embedded SQL naar de programmacode. In het **z/OS perspectief** wordt de COBOL of PL/1 source code aangemaakt, en via een eenvoudige copy/paste wordt de gebruikte en geteste SQL overgenomen in het programma:

```
EXEC SQL
  SELECT TBACCAD.TUTPERSONS.PNO, TBACCAD.TUTPERSONS.PLNAME,
         TBACCAD.TUTPERSONS.PFNAME, TBACCAD.TUTCOMPANIES.CONAME,
         TBACCAD.TUTCOMPANIES.COTOWN, TBACCAD.TUTCOMPANIES.CONO
  FROM  TBACCAD.TUTPERSONS JOIN TBACCAD.TUTCOMPANIES ON
         TBACCAD.TUTPERSONS.PA_CONO = TBACCAD.TUTCOMPANIES.CONO
END-EXEC
```

Enkele aanpassingen of toevoegingen van een `WHERE` clause of andere kunnen natuurlijk zonder probleem gebeuren.

De gegenereerde DCLGEN copybooks worden natuurlijk vooraf met een `EXEC SQL INCLUDE ....  END-EXEC` ingelast.

Het genereren van de load module en vervolgens testen van het programma met embedded SQL gebeurt in RDz natuurlijk op dezelfde manier als programma's zonder databasetoegang, mits het gebruik van de juiste JCL procedure met geïntegreerde DB2 precompiler. Bovendien moeten er binnen de RDz projecten dan nog de correcte *property groups* beschikbaar zijn voor de ontwikkelaar. Maar dit laatste is natuurlijk de verantwoordelijkheid van de RDz administrator.

Maar als je ook het specifieke data access verhaal wilt doorgronden, bestaan er natuurlijk de extra faciliteiten van het **Database Debug perspectief**.

Tot slot

In vergelijking met de ISPF SPUFI omgeving, is het gebruik van de RDz datatools in combinatie met COBOL of PL/1 applicatie-ontwikkeling een stuk gemakkelijker. Binnen dezelfde context kunnen zowel de queries geschreven of gegenereerd en getest worden, en vervolgens naadloos opgenomen in de applicatiecode.

DB2 LUW V10 hidden treasures - administratieve views en routines

Koen De Backer (ABIS)

1. Administratieve objecten: waarover gaat het?

Administratieve objecten in DB2 voor Linux, Unix en Windows (LUW) leveren een gemakkelijke interface voor het aansturen van DB2 beheerstaken en voor het bekijken van performance-gegevens.

De term *objecten* kan in dit geval verwijzen naar:

- tabellen en scalaire functies
- views (vaak gebaseerd op die scalaire functies)
- stored procedures (toegepast in wat men *administratieve routines* noemt)

De gebruikte interface is de gekende, overzichtelijke SQL-interface. Op deze manier kunnen de verschillende tool- en context-specifieke interfaces vermeden worden. Klassieke bronnen en tools voor DB2 LUW configuratie-data, systeemdata, status- en event-data zijn de volgende:

- DB2 profiel: `db2set`
- DB2 configuratie: `db2 get db(m) cfg`
- DB2 cataloog: `select * from syscat.tables, sysstat.tables, ...`
- DB2 `list` command: databases, nodes, tablespaces, containers, utilities ...
- `db2top`, `db2pd`, `db2diag`, `db2mtrk`, ...
- DB2 snapshot monitoring: `db2 get snapshot` (op basis van monitor switches)
- DB2 system - software - licentie ... info
- OS/HW level info

Het gebruik van een aantal van deze tools wordt op dit ogenblik zelfs afgeraden, met name een aantal `list` command arguments, `db2mtrk`, en de snapshot monitor.

Schemanamen worden gebruikt om de administratieve objecten te identificeren:

- `SYSPROC` voor procedures en tabelfuncties
- `SYSIBMADM` voor views die basisinformatie aandragen

Een voorbeeld:

Het alternatief voor

```
get dbm cfg
```

op basis van een tabelfunctie:

```
select * from table(sysproc.dbm_get_cfg()) as config
```

op basis van een view (met als definitie de bovenstaande query):

```
select * from sysibmadm.dbmcfg
```

Overwegingen:

- Niet voor alle tabelfuncties is er een view beschikbaar. Daarenboven kan het in bepaalde situaties aan te raden zijn om de tabelfunctie te gebruiken zelfs al bestaat er een overeenkomstige view. Views zijn "ready to use", functies daarentegen worden uitgevoerd op basis van inputparameters.
- - niet alle administratieve views zijn direct afgeleid van een tabelfunctie:
  - berekeningen kunnen geïntegreerd zijn
  - soms wordt catalooginfo gecombineerd met info verkregen via tabelfuncties
  - soms worden andere administratieve views uitgebreid of gecombineerd
- Databaseobjecten zoals views en functies laten ook een meer gedetailleerde vorm van beveiliging toe. Rechten op objecten kunnen correcter en gemakkelijker toegekend worden aan de betrokken gebruikers. Volgende rechten spelen een rol:
  - voor "`sysproc`"-routine-objecten is "execute" vereist [`SYSCAT.ROUTINEAUTH`]
  - voor "`sysibmadm`"-view-objecten is "select" vereist [`SYSCAT.TABAUT`]

2. De administratieve objecten in vogelvlucht

2.1 Objecten die de DB2 LUW-context beschrijven

Deze objecten geven informatie over het besturingssysteem (eigenschappen en setup), alsook de beschikbare "resources", DB2 software-eigenschappen en geïnstalleerde opties, licentietype, ...

Men heeft het over "environment" views/procedures; ze dragen vaak `*ENV_*` in de naam:

- views: `ENV_CF_SYS_RESOURCES, ENV_FEATURE_INFO, ENV_INST_INFO, ENV_PROD_INFO, ENV_SYS_INFO, ENV_SYS_RESOURCES, ...`

  `DBMCFG, DBCFG, ...`
- procedures: `GET_DB2_SYSTEM_RESOURCES, GET_DB2_EDU_SYSTEM_RESOURCES, GET_ENV_GET_REG_VARIABLES, ...`

2.2 Objecten die bij “monitoring” helpen

“Monitoring” tabelfuncties gebruiken de nieuwe, lichtere en snellere “monitoring” infrastructuur ingevoerd sinds DB2 9.7. De oudere “snapshot monitoring” infrastructuur wordt niet meer opgevolgd. Een van de vernieuwingen is dat de configuratie van de “monitoring” context per database gebeurt en/of via de WLM interface kan worden gestuurd.

“Monitoring” tabelfuncties genereren informatie op 3 niveaus: systeem, activiteit en data object. Daarbij wordt respectievelijk data gebruikt van “request”, “activity” en “data object” monitor elementen. Het gedrag van deze monitor-elementen kan via de databaseconfiguratie worden gestuurd.

Volgende groepen kunnen worden onderscheiden:

- *Systeem Monitoring*: Het systeem-perspectief omvat informatie over alle werk verzet door de database om “requests” van de toepassingen te verwerken. Het zijn de “request”-monitors die de nodige data verzamelen. Systeem monitoring info kan benaderd worden per unit of work, workload, service class, of connection.

Tabelfuncties voor gemonitorde systeeminformatie worden geleverd in paren. De ene functie laat relationele access toe op de verzamelde data, terwijl de andere functie de verzamelde data in een XML-document weergeeft.

```
MON_GET_UNIT_OF_WORK       MON_GET_UNIT_OF_WORK_DETAILS
MON_GET_WORKLOAD           MON_GET_WORKLOAD_DETAILS
MON_GET_SERVICE_SUBCLASS   MON_GET_SERVICE_SUBCLASS_DETAILS
MON_GET_CONNECTION         MON_GET_CONNECTION_DETAILS
MON_GET_AGENT              MON_GET_UTILITY
```

- *Activiteitsmonitoring*: In deze context wordt de klemtoon gelegd op die data die verband houden met het uitvoeren van de kerntaken (“SQL statement execution” in het bijzonder). Monitors verzamelen data op basis van geheugen objecten en de package cache voor activiteiten en SQL instructies. Volgende functies zijn daar een voorbeeld van:

```
MON_GET_ACTIVITY(_DETAILS)
MON_GET_PKG_CACHE_STMT(_DETAILS)
```

- *Data object monitoring:* in deze categorie worden data gebruikt (object monitors) over bewerkingen op database objecten: tabellen, indexen, buffer pools, table spaces, en containers. De object monitors registreren per object elke bewerking op dat object aangevraagd door een request. De volgende functies gebruiken deze verzamelde gegevens:

```
MON_GET_BUFFERPOOL         MON_GET_TABLESPACE
MON_GET_CONTAINER          MON_GET_TABLE
MON_GET_INDEX
```

- Er zijn ook “monitoring” functies die informatie opleveren over locks, system memory en routines. In tegenstelling tot de drie voorgaande groepen is de data voor deze informatie altijd beschikbaar:

```
MON_GET_LOCKS              MON_GET_APPL_LOCKWAIT
MON_GET_MEMORY_SET         MON_GET_MEMORY_POOL
MON_GET_ROUTINE            MON_GET_ROUTINE_EXEC_LIST
MON_GET_ROUTINE_DETAILS    MON_GET_SECTION_ROUTINE
```

Ook voor monitoring is het gebruik van administratieve views mogelijk. Deze views hebben als schema `SYSIBMADM` en hun naam begint meestal met `MON`. Voor functies is dat respectievelijk `SYSPROC` en `MON_GET`. Een paar voorbeelden van views gebaseerd op de nieuwe monitoring-infrastructuur:

```
MON_BP_UTILIZATION         MON_LOCKWAITS
MON_CONNECTION_SUMMARY     MON_PKG_CACHE_SUMMARY
MON_CURRENT_SQL            MON_SERVICE_SUBCLASS_SUMMARY
MON_CURRENT_UOW            MON_TBSP_UTILIZATION
MON_DB_SUMMARY             MON_WORKLOAD_SUMMARY
...
```

2.3 Nuttige info in een pure database-beheerscontext

Voorbeeld: bepalen van tabel dimensies of geheugen verbruik binnen de database), kan opgehaald worden met functies die met "`admin_get`" beginnen of views waarvan de naam "`admin`" vooraan heeft.

```
ADMIN_EST_INLINE_LENGTH
ADMIN_GET_INDEX_COMPRESS_INFO
ADMIN_GET_INDEX_INFO
ADMIN_GET_INTRA_PARALLEL (scalar)
ADMIN_GET_MEM_USAGE
ADMIN_GET_MSGS
ADMIN_GET_STORAGE_PATHS
ADMIN_GET_TAB_COMPRESS_INFO
ADMIN_GET_TAB_DICTIONARY_INFO
ADMINTABINFO / ADMIN_GET_TAB_INFO
ADMINTEMPCOLUMNS / ADMIN_GET_TEMP_COLUMNS
ADMINTEMPTABLES / ADMIN_GET_TEMP_TABLES
ADMIN_IS_INLINED
```

2.4 Procedures die administratieve taken ondersteunen

Met deze procedures kan de beheerder beheerstaken uitvoeren als SQL-opdrachten. Voordelen kunnen zijn:

- de externe APIs worden niet gebruikt (zoals voor de command tools)
- complexe, uit meerdere opdrachten bestaande bewerkingen kunnen met één "call" worden uitgevoerd

`ADMIN_CMD` procedure - administratieve opdrachten uitvoeren

- niet alle opdrachten zijn mogelijk of nuttig
- sommige opdrachten worden uitgevoerd in aangepaste (soms beperktere) vorm

- huidige lijst:

```
ADD CONTACT
ADD CONTACTGROUP
AUTOCONFIGURE
BACKUP - online only
DESCRIBE
DROP CONTACT
DROP CONTACTGROUP
EXPORT
FORCE APPLICATION
IMPORT
INITIALIZE TAPE
LOAD
PRUNE HISTORY/LOGFILE
QUIESCE DATABASE
QUIESCE TABLESPACES FOR TABLE
REDISTRIBUTE
REORG INDEXES/TABLE
RESET ALERT CONFIGURATION
RESET DATABASE CONFIGURATION
RESET DATABASE MANAGER CONFIGURATION
REWIND TAPE
RUNSTATS
SET TAPE POSITION
UNQUIESCE DATABASE
UPDATE ALERT CONFIGURATION
UPDATE CONTACT
UPDATE CONTACTGROUP
UPDATE DATABASE CONFIGURATION
UPDATE DATABASE MANAGER CONFIGURATION
UPDATE HEALTH NOTIFICATION CONTACT LIST
UPDATE HISTORY
```

ADMIN_COPY_SCHEMA procedure - schema's kopiëren
ADMIN_DROP_SCHEMA procedure - schema's verwijderen
ADMIN_MOVE_TABLE procedure - tabellen verplaatsen (online)
ADMIN_MOVE_TABLE_UTIL procedure - vorige procedure aanpassen
ADMIN_REMOVE_MSGS procedure
- verwijdert "data movement" berichten van ADMIN_CMD
ADMIN_REVALIDATE_DB_OBJECTS procedure
- Hervalideert ongeldig geworden database-objecten
ADMIN_SET_INTRA_PARALLEL procedure
- Activeert/desactiveert intra-partitie-parallelisme
ADMIN_SET_MAINT_MODE procedure
- onderhoudsmode voor SQL-compilatie
REORGCHCK_IX_STATS procedure - verfieert opslagkwaliteit
REORGCHCK_TB_STATS procedure - idem
GET_SYSTEM_INFO procedure - configuratie lezen en wijzigen
GET_CONFIG procedure - configuratie lezen en wijzigen
SET_CONFIG procedure - configuratie lezen en wijzigen

3.Toepassingsgebieden: eenvoudigere monitoring, efficiënter problemen opvolgen

Laten we om af te ronden enkele scenario's voorstellen waarbij het gebruik van deze administratieve objecten nuttig kan zijn.

Een eerste scenario kan een "quick-win analysis" worden genoemd.

- zorg ervoor dat je vertrouwd raakt met een beperkt aantal essentiële objecten, zodat prestatiegegevens snel kunnen worden opgehaald en geanalyseerd, "just in case"
- dit laat een onmiddellijke evaluatie van ad-hoc waargenomen prestatievragen toe

In een tweede scenario staat vooral de security centraal.

- toegang tot systeem-, object- en prestatie-data kan worden geregeld a rato van de noden en gebaseerd op standaard "grant"s
- nieuwe views op administratieve views of functies of administratieve views gecombineerd met functies kunnen worden gemaakt
- toegang tot administratieve informatie kan nu worden gekoppeld aan "business" of opdracht specifieke vereisten (denk aan gedetailleerde logging of auditing functies)

Een derde benadering is een scenario gebaseerd op het idee van zogenaamde "management packs"

- maak een reeks eigen ("private") views, procedures en functies gekoppeld aan een "private" beheers schema
- installeer het schema en zijn objecten in alle databases die beheerd moeten worden
- zorg op die manier voor een uniforme, geïntegreerde en soepele interface over de systemen heen voor het detecteren en ontmijnen van "bottlenecks".

Tenslotte kan een scenario worden toegevoegd dat alles bij elkaar brengt in wat je een "performance infrastructuur" kunt noemen.

- bouw een centrale "repository" uit, waar je de "volatiele" data van de administratieve objecten en de management functies opslaat.
  - sla te verzamelde data op in op views gebaseerde tabellen
  - haal de data op geregelde tijdstippen op (vb.: éénmaal per uur)
  - export/import naar een centrale database
  - of, als alternatief, dump de opgehaalde data via een ODBC-connectie in een spreadsheet
- combineer, zoals eerder gesuggereerd, met het "management pack" en zorg zo voor systeem-overkoepelende standaards voor benamingen, timing en datacollecties
- zorg op basis van de opgezette infrastructuur voor systeem- of tijd- gebaseerde rapportering en analyse:
  - op deze wijze ontstaat historische kennis over de database
  - die altijd bruikbaar is voor het bepalen van een "baseline"
  - en tendensen in de ontwikkeling van de database en zijn gebruik documenteert.

CURSUSPLANNING MEI – JUNI 2014

DB2 for z/OS, een totaaloverzicht	2175 EUR	19.05(W)
DB2 UDB for LUW, totaaloverzicht	2175 EUR	19.05(W)
RDBMS-concepten	405 EUR	19.05(W), 23.06(L)
Basiskennis SQL	405 EUR	08.05(L), 20.05(W), 24.06(L)
DB2 for z/OS basiscursus	1365 EUR	26.05(W)
DB2 UDB for LUW basiscursus	1365 EUR	26.05(W)
SQL-QMF voor eindgebruikers	1365 EUR	op aanvraag
SQL workshop	860 EUR	16.06(W), 03.07(L)
SQL voor gevorderden	480 EUR	07.05(L), 21.05(W)
Software-ontwikkeling met SQL PL	960 EUR	op aanvraag
DB2 triggers, stored procedures, UDFs	480 EUR	op aanvraag
DB2 for z/OS programmeren voor gevorderden		op aanvraag
DB2 for z/OS: SQL performance	1440 EUR	op aanvraag
XML in DB2		op aanvraag
DB2 for z/OS database administratie	2020 EUR	op aanvraag
DB2 for z/OS installation and migration	850 GBP	op aanvraag
DB2 for z/OS data recovery	825 GBP	op aanvraag
DB2 for z/OS systems performance and tuning	850 GBP	op aanvraag
DB2 LUW DBA – Kernvaardigheden	1920 EUR	10.06(W)
DB2 10 for z/OS: new features		op aanvraag
DB2 10 for LUW: new features		op aanvraag
SQL voor BI	480 EUR	02.06(L)

Plaats: L = Leuven, W = Woerden, UK = High Wycombe (bij Londen);
alle cursussen ook op aanvraag;
voor details en andere cursussen, zie `http://www.abis.be/html/nlTraining.html`