DB2 NATIVE STORED PROCEDURES @ ALLIANZ BE A PRACTICAL CASE
Agenda

- About Allianz
- The environnement
- Stored procedures: quick overview
- The Allianz case: problem – challenge – solution
- Some lessons learn and tips
- The deployment process
- Summary and conclusions
About Allianz

➡️ Allianz BE

- Allianz Belgium (now becoming Allianz Benelux ..) is an insurance company belonging to a Major group.
- The IT Infrastructure is hosted and partially operated by AMOS (Allianz Managed Operations System)
- Development activity has been handed-over to offshore company (IBM – India)
  - Less control on application “quality …”

➡️ Philippe de Backer

- Member of Technical and Application support team (MF + Open)
- Lead of Mainframe related activities with specific focus on DB2
- Responsible of “controlling” MF performances and resources …
The Environment (1/2)

➡️ Mainframe “centric” environment
  - 3 Lpars
    • Sandbox, Test / Integration, UAT/PROD
➡️ Limited MSU resources
  - Challenge to reduce TCO
➡️ “Traditional” environment
  - IMS DB/DC
  - CICS, DB2, Cobol programs
  - Home made interfaces to IMS DB/DC
    • High level of integration and high level of modularity
    • Old style logical interfaces ➔ Physical interfaces
➡️ New application’s developed in .Net
  - Mix of SQL server and calls to MF interfaces and Databases
The Environment (2/2)

➡️ Use of Microsoft HIS (Host Integration Server)
   - To connect to IMS via LU 6.2 (APPC)

➡️ Migration from ACF/2 to RACF just completed
   - DB2 security not yet migrated to RACF!

➡️ Mainframe optimization done at system level
   - DB2 BP
   - WLM settings
   - I/O subsystem (HPAV)

➡️ Now have to focus on application optimization!
Overview of the Infrastructure (1/2)

**SYSA**
- 4 Hour CPU capping: 54 MSU
- z/OS v1.13
  - IMSEXP: IMS TM/DB V11 Production
  - IMSDUP: IMS TM/DB V11 Acceptance
  - CICSPRD1: CICS TS 3.2 (Owns Alex)
  - CICSPRD2: CICS TS 3.2
  - CICSPRD0: CICS TS 3.2
  - CICSP4A1: CICS TS 3.2 Extract ASF
  - PDB2: DB2 10 CM Production

**SYSB**
- 4 Hour CPU capping: 7 MSU
- z/OS v1.13
  - IMSINT: IMS TM/DB V11 Integration
  - CICSDUP: CICS TS 3.2 Acceptance
  - CICSP4D1: CICS TS 3.2 Extract ASF
  - DDB2: DB2 10 CM Acceptance
  - IDB2: DB2 10 CM Integration

**SYSW**
- 4 Hour CPU capping: 2 MSU
- z/OS v1.13
  - IMSTSTW: IMS TM/DB V11 Systems
  - IMSTST: IMS TM/DB V11 Development
  - CICSTSTW: CICS TS 3.2 Systems
  - CICSTSW: CICS TS 3.2 Development
  - TDB2: DB2 10 CM Systems
Overview of the Infrastructure (2/2)

Mainframe Connectivity Overview

- **Application Users**
  - BMP (JCL) IMS & DB2
  - With 3270 emulation

- **User Interface**
  - MFS Services
  - CODISP

- **Business Logic**
  - IMS MONTRCE
  - CALL

- **Logical Interfaces**
  - IMS
  - CALL

- **Physical Interfaces**
  - IMS/DB
  - PCM1800

- **Data & Database Systems**
  - IMS/DB
  - Tables ALEX (VSAM)
  - DB2

- **DB2 Stored Procedures**
  - Customized & Tuned SQL
  - IMS Code Reuse
  - Strict DB2 Only

- **Application Server Cluster**
  - AGF Framework
  - Microsoft SQL Server

- **Browser**
  - With 3270 emulation

- **External Systems**
  - IP-DLC
  - Microsoft SQL Server
  - DB2 GW

- **Transaction Processing**
  - CICS
  - ALEX

- **User Interface**
  - MFS Services
  - BMS Services

- **Database Management**
  - DB2
  - Tables ALEX (VSAM)

- **Integration Points**
  - DB2 GW
  - ODBC

- **Security & Access**
  - IARD Code Reuse
  - Copy of Tables ALEX in DB2 table

- **Customization & Tuning**
  - Customized & Tuned SQL
To summarize this:

- Access to the data (IMS and/or DB2) done via
  - Application specific interface (VCFC04 as in case 1)
  - Common domain specific interfaces
  - Technical physical interface (MNCALDLI)

- From the “Open” (.NET):
  - APPC (HIS) to call IMS transaction
  - Direct SQL (DB2 connect gateway)
  - Call to COBOL stored procedures (DDF)
  - And even worse: .Net code invoking 3270Emulator via Vbscript to simulate connection to CICS …

- We need to optimize the interfaces to common data!
Stored procedures: the basic picture

Embedded SQL

CALL SP
## DB2 Supported Stored Procedure Languages

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>External High Level Lang. COBOL/C</th>
<th>External High Level Lang. Java</th>
<th>External SQL Stored Procedures</th>
<th>Native SQL Stored Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>COBOL/C Program</td>
<td>Java Program</td>
<td>Generates C Program</td>
<td>Native SQL (SQL PL)</td>
</tr>
<tr>
<td>Computational Complete</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
<td>Yes*</td>
</tr>
<tr>
<td>Integration with SQL</td>
<td>No</td>
<td>No</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Execution Container</td>
<td>WLM SPAS</td>
<td>WLM SPAS</td>
<td>WLM SPAS</td>
<td>Application TCB or DDF**</td>
</tr>
<tr>
<td>Exploit Specialty Engine</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes if DDF**</td>
</tr>
<tr>
<td>Reference External Files</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

*This does not imply that SQL PL has all the features of a high level language

**Executes in DDF address space if executed via distributed access.
External versus Native SQL SP Execution

TIP: Native SP runs under DBM1 ZIIP enabled only if DDF
# Some Native SQL SP Benefits

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| Faster development, increased productivity | • Reduces development effort  
  - Concise language, extended SQL support  
  - All code maintained in DBMS  
• Eliminates language impedance mismatch  
  - Matched data types, integrated with SQL  
• Simplified deployment  
  - No Compiler, no Bind, no WLM |
| Improved performance and lower CPU cost | • Executes in DB2 engine (DBM1)  
  - Avoids overhead of scheduling WLM task  
  - Should perform better than External SQL SP  
• z/OS eligible when executed distributed using DDF |
| Portable code                     | • Implementation of the SQL PSM standard  
• Supported by other RDBMS products, like DB2 for LUW |
Versioning

- Enables multiple versions to be maintained
- A stored procedure must retain the same parameter signature

Maintaining versions
- ALTER ADD VERSION
- ALTER REPLACE VERSION
- ALTER REGENERATE VERSION
- ALTER ACTIVATE VERSION
- ALTER DROP VERSION

Deployment
- Bind deploy

Testing
- SET CURRENT ROUTINE VERSION
Autonomous Transactions

- Autonomous Transaction is a native SQL Procedure that can commit work **OUTSIDE** the commit scope of the calling program
- Executes independently from the calling application
  - Always commits its updates before returning to the caller
  - But does not commit changes in the calling application
  - Useful for event or audit logs

```sql
INSERT INTO T1;
CALL SP1 (autonomous);
  UPDATE T2;
ROLLBACK;
```

```sql
>-----CREATE PROCEDURE---procedure-name------------------------------------------>
  option-list:
    .---COMMIT ON RETURN NO---.
    >--------------------------->
      |---COMMIT ON RETURN YES--|
    '---AUTONOMOUS-------------'
```
THE ALLIANZ CASE
Background ...

Background: Evolution of the MF capacity
- Since January 2011 the downgrade steps are:

- Capacity has been reduced by 50 Mips end October 2012. The downgraded capacity per LPAR's is:

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>MIPS</th>
<th>TARGET CPU BASED ON WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSA</td>
<td>510</td>
<td>60 MSU</td>
</tr>
<tr>
<td>STSB</td>
<td>60</td>
<td>7 MSU</td>
</tr>
<tr>
<td>SySw</td>
<td>17</td>
<td>2 MSU</td>
</tr>
</tbody>
</table>

Where to save CPU?
Issues with the current situation

- Old style application design → lots of interface layers
- Old IMS physical interfaces amended to support DB2
  - Additional layers to transform IMS calls into SQL
- Access the RG (Répertoire général = Customer repository) done via Logical interface
  - Path to the real engine call (SQL) is long … very long
- The central customer repository became a bottleneck
  - IMS DB migrated to DB2
    - Highly integrated in existing applications
- Specific application security managed by the application within the interfaces “path”
- All this is causing high CPU usage
  - Increases occurrence of capping at peak activity period
The case: Access to central customer data (in DB2)

Heaviest processes (batch jobs in this case) are using the same (old) interfaces to RG (Répertoire général)!
The case: Most used DB2 packages (24H)

<table>
<thead>
<tr>
<th>PROGRAMNAME</th>
<th>#_EXEC</th>
<th>_SQL</th>
<th>CPU</th>
<th>ELAPSED</th>
<th>WAITS</th>
<th>CPU%</th>
<th>elapsed/SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYCSTAT</td>
<td>144860</td>
<td>289902</td>
<td>18,694</td>
<td>22,357</td>
<td>0,488</td>
<td>1,81%</td>
<td>0,08</td>
</tr>
<tr>
<td>RGEPG0</td>
<td>87405</td>
<td>806613</td>
<td>30,169</td>
<td>685,463</td>
<td>466,225</td>
<td>2,92%</td>
<td>0,85</td>
</tr>
<tr>
<td>RGEPNM0</td>
<td>85308</td>
<td>834577</td>
<td>27,529</td>
<td>646,85</td>
<td>485,016</td>
<td>2,67%</td>
<td>0,78</td>
</tr>
<tr>
<td>RGEPAD0</td>
<td>85184</td>
<td>776383</td>
<td>31,66</td>
<td>533,213</td>
<td>393,589</td>
<td>3,07%</td>
<td>0,69</td>
</tr>
<tr>
<td>RGEPAC0</td>
<td>71688</td>
<td>1091981</td>
<td>21,665</td>
<td>202,43</td>
<td>100,686</td>
<td>2,10%</td>
<td>0,19</td>
</tr>
<tr>
<td>RGEPNA0</td>
<td>67705</td>
<td>231143</td>
<td>15,551</td>
<td>76,743</td>
<td>56,805</td>
<td>1,51%</td>
<td>0,33</td>
</tr>
<tr>
<td>RGEPPM0</td>
<td>63494</td>
<td>350862</td>
<td>13,544</td>
<td>199,391</td>
<td>90,681</td>
<td>1,31%</td>
<td>0,57</td>
</tr>
<tr>
<td>SROCE0</td>
<td>54373</td>
<td>11916650</td>
<td>387,86</td>
<td>562,471</td>
<td>64,255</td>
<td>37,60%</td>
<td>0,05</td>
</tr>
<tr>
<td>SCPEC0</td>
<td>53597</td>
<td>1450396</td>
<td>73,932</td>
<td>98,88</td>
<td>4,793</td>
<td>7,17%</td>
<td>0,07</td>
</tr>
<tr>
<td>SCPOC0</td>
<td>52263</td>
<td>3616040</td>
<td>74,333</td>
<td>116,582</td>
<td>28,304</td>
<td>7,21%</td>
<td>0,03</td>
</tr>
<tr>
<td>PDIFM1</td>
<td>25670</td>
<td>352550</td>
<td>17,206</td>
<td>60,776</td>
<td>38,096</td>
<td>1,67%</td>
<td>0,17</td>
</tr>
<tr>
<td>RGPIA0</td>
<td>19071</td>
<td>689496</td>
<td>14,29</td>
<td>293,105</td>
<td>73,267</td>
<td>1,39%</td>
<td>0,43</td>
</tr>
<tr>
<td>RGPA0</td>
<td>9745</td>
<td>1466017</td>
<td>15,156</td>
<td>392,656</td>
<td>111,618</td>
<td>1,47%</td>
<td>0,27</td>
</tr>
<tr>
<td>INIF01</td>
<td>6977</td>
<td>155464</td>
<td>12,756</td>
<td>55,099</td>
<td>38,39</td>
<td>1,24%</td>
<td>0,35</td>
</tr>
<tr>
<td>RGGR0</td>
<td>1923</td>
<td>57835</td>
<td>53,659</td>
<td>302,23</td>
<td>227,428</td>
<td>5,20%</td>
<td>5,23</td>
</tr>
<tr>
<td>NVZLG</td>
<td>1568</td>
<td>20712</td>
<td>31,174</td>
<td>61,1</td>
<td>18,385</td>
<td>3,02%</td>
<td>2,96</td>
</tr>
<tr>
<td>PDIFMQ</td>
<td>1304</td>
<td>41042</td>
<td>113,555</td>
<td>175,029</td>
<td>18,946</td>
<td>11,01%</td>
<td>4,26</td>
</tr>
<tr>
<td>SCDS0</td>
<td>1081</td>
<td>3243</td>
<td>44,67</td>
<td>58,454</td>
<td>6,416</td>
<td>4,33%</td>
<td>18,02</td>
</tr>
<tr>
<td>CGEN05IG</td>
<td>28</td>
<td>22447</td>
<td>18,203</td>
<td>21,278</td>
<td>0,005</td>
<td>1,76%</td>
<td>0,95</td>
</tr>
<tr>
<td>FMN2D20</td>
<td>25</td>
<td>6027</td>
<td>15,856</td>
<td>59,022</td>
<td>31,227</td>
<td>1,54%</td>
<td>9,79</td>
</tr>
</tbody>
</table>

Total RG: 1031,462

223,223

21,64%

RG packages are heavily used (# executions) – CPU consumption quite good … BUT …
The case: Drill down in the interface “logic”

➡️ The problem is coming from the way the interfaces are invoked!
➡️ Remember the origin of the RG: **IMS DL/1**
➡️ Interface layer has been added to “transform” DL/1 calls into SQL
  - Complex logic – long path to physical interface
  - **MORE CPU IS USED!**

The conversion process needs to maintain a matrix of IMS functions and SQL conversion rules.
The improvement: usage of dedicated SP’s

We investigated the problem and came to the following options:

<table>
<thead>
<tr>
<th>OPTION</th>
<th>POTENTIAL IMPACT</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Keep situation as is – Do nothing</td>
<td>No further CPU usage will be possible – Risk to be capped during peak activity period is high and will increase as from March 2012</td>
<td>X</td>
</tr>
<tr>
<td>2. Identify key processes (cfr sample) , rewrite it and deploy it to be used on ad-hoc basis</td>
<td>Issue causing 60% of overhead in VI-VG-VS and some On-line processes will be improved – 5-6 MSUs reduction per hour (at peak) is the target – Risk of proliferation will decrease</td>
<td>Δ</td>
</tr>
<tr>
<td>3. Review global interface to RG – Rewrite application Interface – translate function into SP when appropriate (business logic)</td>
<td>CPU reduction may be higher (up to 15-20% at peak) – Lot of improvements foreseen in PROLINK and Blue – No more proliferation of costly interfaces – Build strategy for common functions</td>
<td>O</td>
</tr>
</tbody>
</table>

We decided to review the high level interface (VCFC04) and progressively “transform” the process for selected function:

```
VCFC04 MAIN MODULE

IF FUNCTION = 'GETNR'
    THEN CALL FC3LNR USING PARM1,...
    PERFORM TIMS
END IF

IF FUNCTION = 'GETADDR'
    THEN CALL FC3LNR USING PARM1,...
    PERFROM TIMS
END IF

IF FUNCTION = 'GETKEYP'
    THEN EXECSQL
        CALL SP USING ...
        PERFORM TSQl
END IF
.....
```

SELECT RG09_CLE_PERS
INTO NR12
FROM RTGB09 A
,RTGTB21 B
WHERE A.RG09_NO_PERS = B.RG21_NO_PERS
AND B.RG21_NO_DOSSIER = CN99C
AND B.RG21_DIV_DOSSIER = DV99C
AND B.RG21_TY_INTERV = '10'
AND A.RG09_TY_CLE = '0012';
The results after initial optimisation

Other benefits:

➤ No need to change calling applications (interface will be amended by adding additional SP)
➤ All sub-routines are dynamically called: No need to re-linked it!
➤ Newly developed SP might become available for new applications
➤ Can be the foundation for new standard interface to DB2
Guidelines used at Allianz:

- The SP needs to maintain 2 return-codes
  - A functional return-code (internal logic of the SP)
  - The SQLCODE of the last SQL executed
- The SP will also maintain of “function” code
  - To be used for debugging purposes (last function executed within the SP)
- A SP should never commit work (except autonomous)
- The deployment process should be controlled and easy to use
  - it should allow the developer to focus on logic
- Chose the right balance:
  - Single SQL SP vs business function
  - Keep it simple (avoid result set if possible)
Small tip: Single invocation of a SP to insert a DB « record »

- **Scope**: Tables linked by RI
- **Non natural keys**: usage of identity (Hello “Java” …)

CREATE TABLE TDB2.VGSG01T0
  (SQINN DECIMAL(15, 0) NOT NULL GENERATED BY DEFAULT
   (START WITH 1, INCREMENT BY 1, CACHE 20,
    NO CYCLE, NO ORDER,
    MAXVALUE 999999999999999, MINVALUE 1),
   ...)

Goal: insert of parent and child in a single SP – Solution:

```sql
SET SQLOK = 'NOK';
--
-- INSERT STATEMENT INTO TABLE 01
-- SELECT THE IDENTITY VALUE ASSIGNED BY DB2
--
SET CMNDE = 'ISR01';
SELECT SQINN INTO SQINN_01
FROM FINAL TABLE
  (INSERT INTO VGSG01T0
   (SQEXN, LG99D, DCTYC, DCCLC, DCSTC,
    VL01C, CR01C, MJ01C, VR01N)
  VALUES (SQEXN_01, LG99D_01, DCTYC_01, DCCLC_01,
    DCSTC_01, VL01C_01, CR01C_01, MJ01C_01, VR01N_01));
SET SQL99 = SQLCODE;
--

-- INSERT STATEMENT INTO TABLE 02
-- SELECT THE IDENTITY VALUE ASSIGNED BY DB2
--
SET CMNDE = 'ISR12';
SELECT SQRGN INTO SQRGN_12
FROM FINAL TABLE
  (INSERT INTO VGSG12T0
   (SQINN, RGIDN, SGIDN, RLCAC, KBORN,
    AP99D, RGSTC, PRINC, VL12C, MJ12C)
  VALUES (SQINN_01, RGIDN_12, SGIDN_12, RLCAC_12, KBORN_12,
    AP99D_12, RGSTC_12, PRINC_12, VL12C_12, MJ12C_12));
SET SQL99 = SQLCODE;
--

-- SET KEY OF DC IN OUT PARM.
--
SET SQLOK = 'OK';
END%
```
Deployment process = Appl lifecycle

- **SYSB**
  - **TDB2**
    - **CREATE**
      - INITIAL CREATION OF THE STORED PROCEDURE
      - OWNER / DEV PEOPLE
      - DEV PEOPLE NEEDS CREATE RIGHTS
      - TESTING CAN BE DONE VIA DB2 DATA STUDIO
    - **BIND DEPLOY**
      - DEPLOYMENT INITIATED BY DEV PEOPLE
      - GRANTS ARE Managed BY DBA (REQUEST SENT BY MAIL AS PART OF DEPLOYMENT)
      - SP NEEDS TO BE TESTED WITHIN THE APPLICATION (INTEGRATION)

- **IDB2**
  - **BIND DEPLOY**
    - DEPLOYMENT INITIATED BY DEV PEOPLE
    - GRANTS ARE MANAGED BY DBA (REQUEST SENT BY MAIL AS PART OF DEPLOYMENT)
    - SP NEEDS TO BE TESTED WITHIN THE APPLICATION (INTEGRATION)
    - NOT ACCESSIBLE BY DEV PEOPLE

- **SYSA**
  - **DDB2**
    - **BIND DEPLOY**
      - DEPLOYMENT INITIATED BY DEV PEOPLE
      - GRANTS ARE MANAGED BY DBA
    - **BIND DEPLOY**
      - DEPLOYMENT INITIATED BY DBA AS PART OF APPLICATION RELEASE
      - GO/NO GO DECIDED DURING THE CAB AND AFTER FUNCTIONAL AND TECHNICAL VALIDATION
      - GRANTS MANAGED BY DBA
SP deployment: Required setup

- Define your promotion path and ownership by environment
- Define the DB2’s locations and links
- Process should be amended to support versioning (if needed)
- Define tool usage “scope” : DBA or Developers – Amend DB2 rights accordingly
- New function should be created : UPDATE SP
- Implement DB2 security in RACF to simplify the process ...
The Native SP’s management tool

We developed a very simple ISPF application to manage the SP’s

```
OPTION : EDIT  (EDIT, DROP, CREATE, GRANT)
            (DEPLOY, CREALL, SELECT, DADIC) for user W only
PDS-NAME    => ZCOC0.PDS.DB2SP( TESTSP )
RESULT-SETS => 0
EXECUTES    => READ   (READ, UPDATE or CREATE)
ASUTIME     => 10000

For Deploy From/To status :  TI (TI, ID or DP)
For Create             status :  T  (T, I (SYSB) or D, P (SYSA))
```

Initial DB2 environnement

Need to follow application lifecycle

Minimum set Of parameters
The Native SP’s Management tool (Next)

The EDIT function : Creating the “body” of the SP

<table>
<thead>
<tr>
<th>EDIT</th>
<th>ZCOCO.PDS.DB2SP(TESTSP) - 01.00</th>
<th>Columns 00001 00080</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>==&gt;</td>
<td>Scroll ==&gt; CSR</td>
</tr>
</tbody>
</table>

***** ************************************* Top of Data ***********************************************

000100 CREATE PROCEDURE TESTSP (

000200 IN US99C CHARACTER (8),

000600 OUT FLG99C CHAR (1),

000610 OUT ACCTYP CHAR (8),

000700 OUT SQL99 INTEGER ,

000800 OUT SQLOK CHAR (5))

000900 --

001000 -- STORED PROCEDURE USED TO QUERY THE DB2 RACF TABLES AND CHECK IF

001010 -- USER DOES HAVE ACCESS TO "SECTEUR BOIS" CONTRACTS DATA

001100 --

001200 BEGIN

001300 DECLARE SQLCODE INTEGER ;

001400 DECLARE TYPACC CHAR(8) ;

001500 --

001600 -- DECLARE A GENERAL SQL ERROR EXCEPTION PARAMETER

001700 --

001800 DECLARE CONTINUE HANDLER FOR NOT FOUND

001900 SET SQL99 = SQLCODE ;

002000 DECLARE EXIT HANDLER FOR SQLEXCEPTION

002100 SET SQL99 = SQLCODE ;

002200 --

002300 SET SQLOK = 'NOK' ;

---

- The EDIT function can open a TEMPLATE
- Only the « USER » part of the SP code should be edited
- The IN and OUT parameters needs to have same characteristics as the in caller PGM
- We decided to maintain 2 SQL codes: the one of the SP code itself and the one of the last imbedded SQL statement executed

Basic errors handling
The Native SP’s Management tool (Next)

The EDIT function: Creating the “body” of the SP

EDIT ZCOCO.PDS.DB2SP(TESTSP) - 01.01
Command ==> Columns 00001 00080
Scroll ==> CSR

```
002400  SET SQL99 = 0 ;
002410  SET ACCTYP = ' ' ;
002420  SET FLG99C = ' ' ;
002500  --
002600  -- QUERY THE RACE.DB2 TABLE TO CHECK IF USER DOES HAVE ACCESS
002700  --
002800  SELECT
002900  INTO
002920  TYPACC
003000  FROM USYS.GEMR_ACCESS
003100  WHERE GRACC_CLASS_NAME = 'SECTEUR'
003200  AND GRACC_NAME = 'VGTSECA'
003300  AND GRACC_AUTH_ID = US99C
003500  WITH UR;
003600  ;
004500  --
004510  SET ACCTYP = TYPACC ;
004520  --
004600  IF ACCTYP = 'ALTER' THEN
004700  SET SQL99 = 0 ;
004710  SET FLG99C = '1';
004810  ELSE
```

```
004820  SET SQL99 = 100 ;
004830  SET FLG99C = '0';
004840  END IF;
004900  --
005100  -- RETURN TO CALLER
005200  --
005300  SET SQLOK = 'OK';
005310  SET SQL99 = 0;
005400  END%
```

Simple SQL – No result set to caller
The Native SP’s Management tool (Next)
The DROP function: Ensure the SP does not already exist:

```
OPTION: DROP (EDIT, DROP, CREATE, GRANT)
(DEPLOY, CREALL, SELECT, DADIC) for user W only

PDS-NAME => ZCOCO.PDS.DB2SP( TESTSP )
RESULT-SETS => 0
EXECUTES => READ (READ, UPDATE or CREATE)
ASUTIME => 10000

For Deploy From/To status: TI (TI, ID or DP)
For Create status: T (T, I (SYSB) or D, P (SYSA))
```
The Native SP’s Management tool (Next)

The CREATE function : Creating the SP in DB2

OPTION : CREATE (EDIT, DROP, CREATE, GRANT) (DEPLOY, CREALL, SELECT, DADIC) for user W only

PDS-NAME => ZCOCO.PDS.DB2SP( TESTSP )
RESULT-SETS => 0
EXECUTES => READ (READ, UPDATE or CREATE)
ASUTIME => 10000

For Deploy From/To status : TI (TI, ID or DP)
For Create status : T (T, I (SYSB) or D, P (SYSA))

• Technical part of the SP definition has been added by the REXXX
• Only the basic parameters provided in the initial screen are used to build the SP :
  • NO DYNAMIC RESULT SET
  • READ ONLY
  • COMMIT ON RETURN is NO by default
How to Re-create a SP and keep existing rights (EXECUTION) alive? CREALL function

The CREALL function will:

- Make a select on SYSIBM.SYSPROUTINEAUTH to retrieve the current GRANTEE
- DROP the SP
- CREATE it again
- GRANT it EXECUTE rights again

**Input Statement:**

GRANT EXECUTE ON PROCEDURE TDB2.TESTSP TO WISERV, WJDEVX

**Result of SQL Statement:**

DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
DSNT418I SQLSTATE = 0000 SQLSTATE RETURN CODE
DSNT416I SQLERRD = 0 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD = X'00000000' X'00000000' X'00000000' X'00000000' X'00000000' SQL DIAGNOSTIC INFORMATION

GRANT SUCCESSFUL
The Native SP’s Management tool (Next)
The DEPLOY function: promotion to the next environment

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DEPLOY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(EDIT, DROP, CREATE, GRANT)</td>
</tr>
<tr>
<td></td>
<td>(DEPLOY, CREALL, SELECT, DADIC) for user W only</td>
</tr>
</tbody>
</table>

- **PDS-NAME** => ZCOCO.PDS.DB2SP( TESTSP )
- **RESULT-SETS** => 0
- **EXECUTES** => READ (READ, UPDATE or CREATE)
- **ASUTIME** => 10000

For Deploy From/To status: DP (TI, ID or DP)
For Create status: T (T, I (SYSB) or D, P (SYSA))

➔ A batch job will be submitted on the LPAR where the origin DB2 is running
➔ The appropriate setup needs to be done to allow connection between the DB2 by following the application lifecycle (promotion path)
➔ Owner and qualifier are generated based on the target DB2
➔ Mail sent to DBA to ask for the GRANTS (as from IDB2)
The Native SP’s Management tool (Next)

The DEPLOY function : promotion to the next environment

```plaintext
//WPDBCKDE JOB (09010,ORD1), 'Deploy TESTSP',
// NOTIFY=WPDBCK, MSGLEVEL=(1,1), CLASS=A
/*JOB...*/
//BNDD1EX EXEC TSOBATCH
//DBRMLIB DD DSN=TLIB.IDB2.DBRMLIB,DISP=SHR
// DD DSN=TLIB.DDB2.DBRMLIB,DISP=SHR
//SYSTSIN DD *

DSN SYSTEM(IDB2)
   BIND PACKAGE(DDB2.DDB2) -
   OWNER(DDB2) -
   QUALIFIER(DDB2) -
   COPYVER(V1) -
   DEPLOY(IDB2.TESTSP) -
   ACTION(REP)

END
/*

IEF375I JOB/WPDBCKDE/START 2013318.1629
IEF033I JOB/WPDBCKDE/STOP 2013318.1629
   CPU:  0 HR  00 MIN 00.02 SEC  SRB:  0 HR  00 MIN 00.00 SEC
READY
DSN SYSTEM(IDB2)
  DSN
     BIND PACKAGE(DDB2.DDB2)  OWNER(DDB2)  QUALIFIER(DDB2) COPYVER(V1) DEPLOY(IDB2.TESTSP) ACTION(REP)
  DSN
     SUCCESSFUL BIND FOR PACKAGE = DDB2.DDB2.TESTSP.(V1)
DSNT232I }
  ENDDSN
  READY
  END
DATA SET UTILITY - GENERATE
PROCESSING ENDED AT EOD
```
SUMMARY AND CONCLUSIONS
This was our Agenda

- About Allianz
- The environnement
- Stored procedures: quick overview
- The Allianz case: problem – challenge – solution
- Some lessons learn and tips
- The deployment process
- Summary and conclusions
Take aways

➡️ Mayor issues:
   – Deployment process

➡️ Mayor benefits:
   – Better performance
   – Simplification
   – Centralisation of code = code reuse

➡️ Next steps
   – zIIP
   – Versioning
Questions?

THANKS!

PHILIPPE DE BACKER
Philippe.de_backer@allianz.be