Agenda

• IFCID 366 Introduction
• System and Configuration
  • System State before DB2 v11R1
  • Starting I366 trace
• Externalize I366 and SQL text
  • EXPLAIN_STMT_CACHE/Unload SYSPACKSTMT
  • I366 TRACE FROM SMF
• Reporting and Analysis (with IDAA)
• Application Testing and Implementation
• Be Aware….  
• Questions
IFCID 366 Introduction

• IFCID 366 is a record trace that identifies V10 incompatible functions.
• IFCID 366 was first introduced in DB2 V10 as a method to identify (CAST) CHAR, (CAST) VARCHAR and TIMESTAMP application incompatibilities.
• DB2 V11 enhanced I366 with additional tracing capabilities.
• DB2 V11 introduces a second IFCID, IFCID 376. IFCID 376 is a roll up of 366. DB2 writes one record for each unique static or dynamic statement.
• I366 Function Code Incompatibilities are explained on IBM Knowledge Center: https://www.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/wnew/src/tpc/db2z_whatschanged.html
System and Configuration

- System State
  - 2 Production Systems and 4 Non-Production Systems
  - DB2 v10 with BIF_COMPATABILITY = V9_DECIMALVARCHAR
  - “The BIF_COMPATIBILITY subsystem parameter specifies whether built-in functions and specifications are to return results in the newer format or revert to the pre-DB2® 10 format.”

- Start Command IFCID 366
  - -START TRACE(PERFM) CLASS(32) IFCID(366) SCOPE (GROUP) DEST(SMF)
  - Reminder to start trace after system cycle, especially data sharing scope: Group
Overall Process Flow

- Tables with I366 trace and SQL text of Dynamic and Static Statements

1. Explain STMTCACHE/UNLOAD SYSPACKSTMT
2. HPU
3. MD5 Hash/Unique SQL
4. Extract and Load IFCID I366 STMTS
EXPLAIN STMTCACHE/Unload SYSPACKSTMT

- “EXPLAIN STMTCACHE ALL” for every DB2 member

- Unload SYSPACKSTMT – Load to STATIC_STG table

- Executed once per day at midnight

- Execute HPU Unload of DSN_STATEMENT_CACHE_TABLE to flat file using SELECT statement
  - CAST(SUBSTR(STMT_TEXT,1,8000) AS VARCHAR(8000))
  - IDAA does not handle LOB fields

- For each STMTCACHE file, LOAD to STMT_CACHE_STG table

- For SYSPACKGE files - Load to STATIC_STG table
HPU Unload from STMT_CACHE_STG table where the row doesn’t already exist on our STMT_CACHE repository table (identifying new records only):

```sql
UNLOAD TABLESPACE
SELECT
    STG.STMT_ID,
    COALESCE(STG.STMT_TOKEN,'NODS'),
    STG.COLLID,
    STG.PROGRAM_NAME,
    STG.INV_DROPALT,
    STG.STMT_TEXT
FROM STMT_CACHE_STG STG
WHERE NOT EXISTS
( SELECT *
    FROM DB2ADMN.TDB_STMT_CACHE A
    WHERE STG.PROGRAM_NAME = A.PROGRAM_NAME
    AND STG.CACHED_TS = A.CACHED_TS
    AND STG.PRIMAUTH = A.PRIMAUTH
    AND STG.SCHEMA = A.SCHEMA
    AND STG.GROUP_MEMBER = A.GROUP_MEMBER
) WITH UR;
```
HPU Unload from STATIC_STG table where the row doesn’t already exist on our STATIC_SQL repository table (identifying new records only):

WITH PKG_STAT (COLLECTION, PROGRAM, PKG_TIMESTAMP) AS (
    SELECT COLLECTION, PROGRAM, MAX(PKG_TIMESTAMP)
    FROM DB2ADMN.CQM31_53_STATIC_STG_SQL
    GROUP BY COLLECTION, PROGRAM
    MINUS
    SELECT SPKG.COLLECTION, SPKG.PROGRAM, MAX(SPKG.PKG_TIMESTAMP)
    FROM DB2ADMN.CQM31_53_STATIC_SQL SPKG
    WHERE...
)

SELECT

,CAST(SUBSTR(SQLSTMT,1,8000) AS VARCHAR(8000))
FROM DB2ADMN.CQM31_53_STATIC_STG_SQL STGSQL, PKG_STAT
WHERE STGSQL.COLLECTION = PKG_STAT.COLLECTION
AND STGSQL.PROGRAM = PKG_STAT.PROGRAM
AND STGSQL.PKG_TIMESTAMP = PKG_STAT.PKG_TIMESTAMP
MD5 Hash/Unique SQL

- Execute Rexx exec that call a stored procedure
- Reads HPU unloaded file
- Calls Stored Procedure to generate an MD5 hash value for each unique SQL Statement
- Inserts data into two repository tables: STMT_CACHE table and STMT_CACHE_TEXT table
- STMT_CACHE table has all columns except STMT_TEXT
- STMT_CACHE_TEXT has only MD5 hash value and STMT_TEXT
- If we receive -803, we know the SQL Statement is already on our table.
Extract and Load IFCID 366 Records

- Dump TYPE(102) Records from SMF
- Execute SAS program to extract relevant IFCID366 data using VMAC102 and _T102366 macro
- Mapping for IFCID366 records can be found in SDSNMACS(DSNDQW05)
Extract and Load IFCID 366 Records

- IF QWHCATYP = 1 THEN F366TYP = 'TSO ';
- IF QWHCATYP = 2 THEN F366TYP = 'DB2 CALL ATTACH';
- IF QWHCATYP = B THEN F366TYP = 'DB2 UTILITIES';
- IF QWHCATYP = C THEN F366TYP = 'RRSAF';
- IF QWHCATYP = 8 THEN F366TYP = 'DRDA ';
- IF QW0366FN = 1 THEN
  - F366DESC = 'V9 SYSIBM.CHAR(DECIMAL-EXPR) FUNCTION';
- IF QW0366FN = 2 THEN
  - F366DESC = 'V9 SYSIBM.VARCHAR(DECIMAL-EXPR)
  FUNCTION CAST (DECIMAL AS CHAR)';
- IF QW0366FN = 3 THEN F366DESC = 'UNSUPPORTED TIMESTAMP VALUE';
- IF QW0366FN = 4 THEN F366DESC = 'RESERVED WORD:ARRAY_EXISTS';
- IF QW0366FN = 5 THEN F366DESC = 'RESERVED WORD:CUBE';
- IF QW0366FN = 6 THEN F366DESC = 'RESERVED WORD:ROLLUP';
- IF QW0366FN = 7 THEN F366DESC = 'POSSIBLE SQLCODE -301';
Extract and Load IFCID 366 Records

- IF QW0366FN = 8 THEN F366DESC = 'DDF_COMPATIBILITY/JAVA EXCP';
- IF QW0366FN = 9 THEN F366DESC = 'TIMEZONE IGNORE';
- IF QW0366FN = 1101 THEN F366DESC = 'XML NODE';
- IF QW0366FN = 1102 THEN F366DESC = 'XPATH PROCESSING';
- IF QW0366FN = 1103 THEN F366DESC = 'ASUTIME';
- IF QW0366FN = 1104 THEN F366DESC = 'CLIENT_USERID';
- IF QW0366FN = 1105 THEN F366DESC = 'CLIENT WKSTNAME';
- IF QW0366FN = 1106 THEN F366DESC = 'CLIENT_APPLNAME';
- IF QW0366FN = 1107 THEN F366DESC = 'CLIENT_ACCTING';
- IF QW0366FN = 1108 THEN F366DESC = 'CAST(STRING AS TIMESTAMP)';
- IF QW0366FN = 1109 THEN F366DESC = 'CAST(STRING AS TIMESTAMP)';
- IF QW0366FN = 1110 THEN F366DESC = 'BUILT-IN FUNCTION: SPACE';
- IF QW0366FN = 1111 THEN F366DESC = 'BUILT-IN FUNCTION: VARCHAR';

- IF QW0366TY = '4000'X THEN F366TYPE = 'STATIC';
- IF QW0366TY = '8000'X THEN F366TYPE = 'DYNAMIC';
Extract and Load IFCID 366 Records

Fields Extracted from IFCID 366 and Loaded to IFCID 366 table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QWHAMEMN</td>
<td>DB2 Member Name</td>
<td>F366TYPE</td>
<td>‘STATIC’ or ‘DYNAMIC’ based on QH0366TY</td>
</tr>
<tr>
<td>MYDATE</td>
<td>SMF Date</td>
<td>QW0366FN</td>
<td>FunctionCode</td>
</tr>
<tr>
<td>QWHCAID</td>
<td>Authid</td>
<td>F366DESC</td>
<td>Text description of function code</td>
</tr>
<tr>
<td>QWHCCN</td>
<td>Connection Name</td>
<td>QW0366SN</td>
<td>Statement number of query</td>
</tr>
<tr>
<td>QWHCEUWN</td>
<td>End user Workstation Name</td>
<td>QW0366SE</td>
<td>Section number</td>
</tr>
<tr>
<td>F366TYP</td>
<td>Type Code</td>
<td>QW0366SI</td>
<td>Statement Identifier</td>
</tr>
<tr>
<td>QWHCCV</td>
<td>Correlation ID Value</td>
<td>QW0366TS</td>
<td>Timestamp for query</td>
</tr>
<tr>
<td>QWHSSTCK</td>
<td>StoreClock</td>
<td>QW0366VL</td>
<td>Version Length</td>
</tr>
<tr>
<td>QWHSIID</td>
<td>Subsystem Name</td>
<td>QW0366VN</td>
<td>Version name</td>
</tr>
<tr>
<td>QWHDRQNM</td>
<td>Requestor location Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QWHCTOKN</td>
<td>Accounting Token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QW0366PC</td>
<td>Package collection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QW0366PN</td>
<td>Program name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QW0366PL</td>
<td>Plan Name for query</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reporting and Analysis

- Capturing and externalizing IFCID 366 and the corresponding SQL statements allowed us to build reports by function codes, DB2 member, PRIMAUTH, date and application name easily using IDAA.
SELECT I366.MEMBER,
    I366.PRIMAUTH,
    I366.CORRELATION_ID,
    I366.COLLECTION_ID,
    I366.PROGRAM_NAME,
    I366.PLAN_NAME,
    I366.STMT_TYPE,
    I366.FUNCTION_CODE,
    I366.FUNCTION_DESC,
    I366.D_STMT_ID,
    I366.SMF_DATE,
CASE
    WHEN PRIMAUTH LIKE 'DIRECT%' THEN 'FTD'
    WHEN SUBSTR(primauth,1,1) = 'E' THEN 'Person'
    WHEN SUBSTR(primauth,1,1) = 'T' THEN 'Person'
    WHEN SUBSTR(primauth,1,1) = 'S' THEN 'Person'
    WHEN SUBSTR(primauth,1,1) = 'P' THEN SUBSTR(PRIMAUTH,3,2)
    ELSE SUBSTR(PRIMAUTH,2,2)
END AS ApplicationID
FROM     DB2ADMN.TDB_SMF_IFCID366 I366
WHERE
    I366.STMT_TYPE = 'DYNAMIC'
    AND I366.MEMBER LIKE 'DBT%'
    AND I366.FUNCTION_CODE IN (1,2,3,4,5,6,7,8)
    AND I366.PRIMAUTH LIKE 'FDWP%'
GROUP BY I366.MEMBER, I366.PRIMAUTH, I366.CORRELATION_ID, I366.COLLECTION_ID,
    I366.PROGRAM_NAME, I366.PLAN_NAME, I366.STMT_TYPE, I366.FUNCTION_CODE,
    I366.FUNCTION_DESC, I366.D_STMT_ID, I366.SMF_DATE;
Reporting and Analysis: STMT_TEXT

- Analysis required reviewing STMT_TEXT and searching for key items as CASTs (using regular expressions search)

<table>
<thead>
<tr>
<th>TEXT_HASH</th>
<th>STMT_TYPE</th>
<th>FUNCTION_CODE</th>
<th>FUNCTION_DESC</th>
<th>STMT_TEXT</th>
<th>APPLICATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>2 59 SYMBM.VARCHAR(DECIMAL-EXPR)</td>
<td>SELECT STG.I_REC_KEY, STG.I_ORG_UNIT_KEY, STG.I_GL_ACCT,</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_ACCTNG</td>
<td>select CURRENT_CLIENT_ACCTNG, CURRENT_CLIENT_APPNAME, CURRENT_</td>
<td>RT</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>RT</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>UP</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select this.i_approval as i2_25_2, this.i_company as i3_25_2, this.i_expiry as</td>
<td>TFD</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>2 59 SYMBM.VARCHAR(DECIMAL-EXPR)</td>
<td>SELECT SUBSTR(DTLFINACCT '1,15' )</td>
<td>DW</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>2 59 SYMBM.VARCHAR(DECIMAL-EXPR)</td>
<td>SELECT FN.FN_OV_FCT_DISPSTN.OV_FCT_DISPSTN_ID</td>
<td>VA</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>TP</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select this.fraction as strtimestamp.16.CAST(TIMESTAMP(TOSTR(I11111,00)) as</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>2 59 SYMBM.VARCHAR(DECIMAL-EXPR)</td>
<td>SELECT FPV.LC_STX, FPV.LC_VIN, FPV.LCL_UD, FPV.LIDL, FPV.LUDL</td>
<td>FTD</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>3 59 UNSUPPORTED CHARACTER STRING REPRES</td>
<td>WITH STAGE001 AS</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_ACCTNG</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>BP</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>FA</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1107 CLIENT_ACCTNG</td>
<td>select CURRENT_CLIENT_ACCTNG, CURRENT_CLIENT_APPNAME, CURRENT_</td>
<td>RT</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select SESSION_TIME_ZONE, CURRENT_CLIENT_ACCTNG, CURRENT_</td>
<td>RT</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>1104 CLIENT_USERID</td>
<td>select this.i_approval as i2_25_2, this.i_company as i3_25_2, this.i_expiry as</td>
<td>TFD</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>2 59 SYMBM.VARCHAR(DECIMAL-EXPR)</td>
<td>SELECT BT LOLOC00.COLL_GROUP</td>
<td>BT</td>
<td></td>
</tr>
<tr>
<td>09b3f636f</td>
<td>DYNAMIC</td>
<td>3 59 UNSUPPORTED CHARACTER STRING REPRES</td>
<td>Select.TIMESTAMPDIFF(18, CAST(TIMESTAMP(TOSTR(I11111,00)) as</td>
<td>Person</td>
<td></td>
</tr>
</tbody>
</table>
### Reporting and Analysis: SQLSTMT

- Static Statements required manual review of SQLs in Packages

<table>
<thead>
<tr>
<th>STMT_TYPE</th>
<th>FUNCTION</th>
<th>FUNCTION_DESC</th>
<th>SQLSTMT</th>
<th>QUERY_STMT_HDR COLLECTION_ID</th>
<th>PROGRAM_NAME</th>
<th>PLAN_NAME</th>
<th>PRIMA_APPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATIC</td>
<td>1104</td>
<td>CLIENT_USERID</td>
<td>SELECT CURRENT CLIENT_ACCTNG . CURRENT CLIENT_APPNAME . CURRENT..</td>
<td>312 SYSACCEL</td>
<td>ADTOBCON</td>
<td>ADTSCALL</td>
<td>PBDWD .. DW</td>
</tr>
<tr>
<td>STATIC</td>
<td>1107</td>
<td>CLIENT_ACCTNG</td>
<td>SELECT CURRENT CLIENT_ACCTNG . CURRENT CLIENT_APPNAME . CURRENT..</td>
<td>312 SYSACCEL</td>
<td>ADTOBCON</td>
<td>ADTSCALL</td>
<td>PBDWD .. DW</td>
</tr>
<tr>
<td>STATIC</td>
<td>1106</td>
<td>CLIENT_APPNAME</td>
<td>INSERT INTO ACP1_TAC_FUNDTSFS_SECUR_AUD ..</td>
<td>2 ACP1</td>
<td>RAC02301</td>
<td>DISTSERV</td>
<td>FACPA .. AC</td>
</tr>
<tr>
<td>STATIC</td>
<td>1 V9</td>
<td>SYSTM CHAR(DECIMAL-EXPR)</td>
<td>SELECT SUBSTR( CHAR(XL40 CR AS OF DAT ) .. 9, 2 ) INTO : H FROM ..</td>
<td>3214 CBT001B</td>
<td>BTPREDT</td>
<td>CBT001B</td>
<td>PBDTO .. BT</td>
</tr>
<tr>
<td>STATIC</td>
<td>1107</td>
<td>CLIENT_ACCTNG</td>
<td>SELECT CURRENT CLIENT_ACCTNG . CURRENT CLIENT_APPNAME . CURRENT..</td>
<td>308 SYSACCEL</td>
<td>ADTOBCON</td>
<td>ADTSCALL</td>
<td>PBDWD .. DW</td>
</tr>
<tr>
<td>STATIC</td>
<td>1104</td>
<td>CLIENT_USERID</td>
<td>SELECT CURRENT CLIENT_ACCTNG . CURRENT CLIENT_APPNAME . CURRENT..</td>
<td>308 SYSACCEL</td>
<td>ADTOBCON</td>
<td>ADTSCALL</td>
<td>PBDWD .. DW</td>
</tr>
<tr>
<td>STATIC</td>
<td>1105</td>
<td>CLIENT_YSTNAME</td>
<td>SELECT CURRENT CLIENT_ACCTNG . CURRENT CLIENT_APPNAME . CURRENT..</td>
<td>312 SYSACCEL</td>
<td>ADTOBCON</td>
<td>ADTSCALL</td>
<td>TS4300A .. Power</td>
</tr>
</tbody>
</table>
WITH QM_CTE (
  MEMBER, PRIMAUTH, CORRELATION_ID, COLLECTION_ID, PROGRAM_NAME, PLAN_NAME,
  STMT_TYPE, FUNCTION_CODE, FUNCTION_DESC, D_STMT_ID, D_STMT_ID_d
) AS
  (SELECT I366.MEMBER, I366.PRIMAUTH, I366.CORRELATION_ID, I366.COLLECTION_ID, I366.PROGRAM_NAME, I366.PLAN_NAME,
  I366.STMT_TYPE, I366.FUNCTION_CODE, I366.FUNCTION_DESC, I366.D_STMT_ID, i366.smf_date
   FROM DB2ADMN.TDB_SMF_IFCID366 I366
   WHERE I366.STMT_TYPE = 'DYNAMIC'
   AND YEAR(UNIX_TIMESTAMP_FORMAT(I366.SMF_DATE, 'mm/dd/yy')) = 2016
   AND MONTH(UNIX_TIMESTAMP_FORMAT(I366.SMF_DATE, 'mm/dd/yy')) = 8
   AND (I366.member like 'DRP%' OR I366.member like 'DRU%')
   GROUP BY I366.MEMBER,
   I366.PRIMAUTH, I366.CORRELATION_ID, I366.COLLECTION_ID, I366.PROGRAM_NAME,
   I366.PLAN_NAME, I366.STMT_TYPE, I366.FUNCTION_CODE, I366.FUNCTION_DESC,
   I366.D_STMT_ID, I366.SMF_DATE)
,
  STMT AS (SELECT
    QM_CTE.MEMBER,
    QM_CTE.PRIMAUTH, QM_CTE.CORRELATION_ID, QM_CTE.COLLECTION_ID, QM_CTE.PROGRAM_NAME, QM_CTE.PLAN_NAME,
    QM_CTE.STMT_TYPE, QM_CTE.FUNCTION_CODE, QM_CTE.FUNCTION_DESC, QM_CTE.D_STMT_ID,
    TEXT.STMT_TOKEN, TEXT.TEXT_HASH,
    TEXT1.STMT_TEXT
   FROM QM_CTE
   LEFT JOIN db2admn.TDB_STMT_CACHE TEXT
   ON QM_CTE.PROGRAM_NAME = TEXT.PROGRAM_NAME
   AND QM_CTE.PRIMAUTH = TEXT.PRIMAUTH
   AND QM_CTE.MEMBER = TEXT.GROUP_MEMBER
   AND QM_CTE.D_STMT_ID = TEXT(stmt_ID)
   JOIN db2admn.TDB_STMT_CACHE_TEXT TEXT1
   ON TEXT.TEXT_HASH = TEXT1.TEXT_HASH
   WHERE TEXT.CACHED_TS = (SELECT MAX(TEXT2.CACHED_TS) FROM db2admn.TDB_STMT_CACHE TEXT2
   WHERE QM_CTE.PROGRAM_NAME = TEXT2.PROGRAM_NAME
   AND QM_CTE.PRIMAUTH = TEXT2.PRIMAUTH
   AND QM_CTE.MEMBER = TEXT2.GROUP_MEMBER
   AND QM_CTE.D_STMT_ID = TEXT2.stmt_ID
   AND TEXT2.CACHED_TS <= UNIX_TIMESTAMP_FORMAT(d_i366, 'mm/dd/yy') + 1 day)
)
SELECT
  stmt.TEXT_HASH,
  stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
  stmt.STMT_TOKEN, stmt.TEXT_HASH,
  stmt1.STMT_TEXT
FROM stmt
WHERE stmt.TEXT_HASH = stmt1.TEXT_HASH
GROUP BY stmt.TEXT_HASH,
stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
stmt.STMT_TOKEN, stmt.TEXT_HASH,
stmt1.STMT_TEXT,

CASE
  WHEN PRIMAUTH LIKE 'DIRECT%' THEN 'FTD'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'E' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'T' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'S' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'P' THEN SUBSTR(stmt.primauth, 3, 2)
  ELSE SUBSTR(stmt.primauth, 2, 2)
END AS ApplicationID

GROUP BY stmt.TEXT_HASH,
stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
stmt.STMT_TOKEN, stmt.TEXT_HASH,
stmt1.STMT_TEXT,

CASE
  WHEN PRIMAUTH LIKE 'DIRECT%' THEN 'FTD'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'E' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'T' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'S' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'P' THEN SUBSTR(stmt.primauth, 3, 2)
  ELSE SUBSTR(stmt.primauth, 2, 2)
END AS ApplicationID

GROUP BY stmt.TEXT_HASH,
stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
stmt.STMT_TOKEN, stmt.TEXT_HASH,
stmt1.STMT_TEXT,

CASE
  WHEN PRIMAUTH LIKE 'DIRECT%' THEN 'FTD'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'E' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'T' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'S' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'P' THEN SUBSTR(stmt.primauth, 3, 2)
  ELSE SUBSTR(stmt.primauth, 2, 2)
END AS ApplicationID

GROUP BY stmt.TEXT_HASH,
stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
stmt.STMT_TOKEN, stmt.TEXT_HASH,
stmt1.STMT_TEXT,

CASE
  WHEN PRIMAUTH LIKE 'DIRECT%' THEN 'FTD'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'E' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'T' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'S' THEN 'Person'
  WHEN SUBSTR(stmt.primauth, 1, 1) = 'P' THEN SUBSTR(stmt.primauth, 3, 2)
  ELSE SUBSTR(stmt.primauth, 2, 2)
END AS ApplicationID

GROUP BY stmt.TEXT_HASH,
stmt.STMT_TYPE, stmt.FUNCTION_CODE, stmt.FUNCTION_DESC,
stmt.STMT_TOKEN, stmt.TEXT_HASH,
stmt1.STMT_TEXT,
WITH QM_CTE (MEMBER, PRIMAUTH, CORRELATION_ID, COLLECTION_ID, PROGRAM_NAME, PLAN_NAME, STMTCODE, FUNCTION_CODE, FUNCTION_DESC, D_STMT_ID, QUERY_STMT_NBR) AS
FROM DB2ADMN.TDB_SMF_IFCID366 I366
WHERE I366.STMT_TYPE = 'STATIC'
AND YEAR(TIME_STAMP_FORMAT(I366.SMF_DATE, 'mm/dd/yy')) = '2016'
AND MONTH(TIME_STAMP_FORMAT(I366.SMF_DATE, 'mm/dd/yy')) = '8'
AND (I366.MEMBER LIKE 'DRP%' OR I366.MEMBER LIKE 'DRU%' OR I366.MEMBER LIKE 'DDT%' OR I366.MEMBER LIKE 'DDD%')
, STMT as (SELECT QM_CTE.MEMBER, QM_CTE.PRIMAUTH, QM_CTE.CORRELATION_ID, QM_CTE.COLLECTION_ID, QM_CTE.PROGRAM_NAME, QM_CTE.PLAN_NAME, QM_CTE.STMT_TYPE, QM_CTE.FUNCTION_CODE, QM_CTE.FUNCTION_DESC, QM_CTE.D_STMT_ID, TEXT.TEXT_HASH, TEXT1.SQLSTMT
FROM QM_CTE
LEFT JOIN DB2ADMN.CQM31_53_STATIC_SQL TEXT
ON QM_CTE.PROGRAM_NAME = TEXT.PROGRAM
AND QM_CTE.QUERY_STMT_NBR = TEXT_STMT
AND CASE WHEN QM_CTE.MEMBER LIKE 'DBP%' THEN 'PDG1'
WHEN QM_CTE.MEMBER LIKE 'DRP%' THEN 'PRG1'
WHEN QM_CTE.MEMBER LIKE 'DWP%' THEN 'PWG1'
ELSE '' END = TEXT.DB2_SUBSYSTEM
AND QM_CTE.COLLECTION_ID = TEXT_COLLECTION
JOIN DB2ADMN.CQM31_53_STATIC_SQL_STMT TEXT1
ON TEXT.TEXT_HASH = TEXT1.TEXT_HASH
WHERE TEXT_PKG_TIMESTAMP = (SELECT MAX(TEXT2_PKG_TIMESTAMP) FROM DB2ADMN.CQM31_53_STATIC_SQL TEXT2
WHERE QM_CTE.PROGRAM_NAME = TEXT2_PKG_TIMESTAMP
AND QM_CTE.QUERY_STMT_NBR = TEXT2_STMT
AND CASE WHEN QM_CTE.MEMBER LIKE 'DBP%' THEN 'PDG1'
WHEN QM_CTE.MEMBER LIKE 'DRP%' THEN 'PRG1'
WHEN QM_CTE.MEMBER LIKE 'DWP%' THEN 'PWG1'
ELSE '' END = TEXT.DB2_SUBSYSTEM
AND QM_CTE.COLLECTION_ID = TEXT2_COLLECTION
AND TEXT2_PKG_TIMESTAMP <= (TIME_STAMP_FORMAT(I366,'mm/dd/yy') + 1 day)
);
Reporting and Analysis: Communication

• The report allowed us to analyze and test functions to determine impact.
• For example: we found that these functions would not impact us:

<table>
<thead>
<tr>
<th>FC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1104</td>
<td>Long CLIENT_ACCTNG Special Reg</td>
</tr>
<tr>
<td>1105</td>
<td>Long CLIENT_APPLNAME Special Reg</td>
</tr>
<tr>
<td>1106</td>
<td>Long CLIENT_USERID Special Reg</td>
</tr>
<tr>
<td>1107</td>
<td>Long CLIENT_WRKSTNNAME Special Reg</td>
</tr>
</tbody>
</table>

• Focus on functions that will be impacted
• For example: impactful incompatibilities

<table>
<thead>
<tr>
<th>FC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V9 SYSIBM.CHAR(decimal-expr) function</td>
</tr>
<tr>
<td>2</td>
<td>V9 SYSIBM.VARCHAR(decimal-expr) function. CAST (decimal as VARCHAR or CHAR)</td>
</tr>
</tbody>
</table>
Reporting and Analysis: Communication

• Informed the Application Teams of incompatibilities
• Informed them of required code changes and, their testing and implementation options.

• Currently in DB2v11 conversion mode(CM)
• Certain functions/application behavior is different in DB2 v11 New Function Mode(NFM) – incompatible functions
• Application team options –
  — Depending on the type of incompatibility
    — Option #1 - Application code change not required
    — Option #2 - Change code ahead of NFM
    — Option #3 - Change code after migrating to NFM

Behavior Change

• Changed CHAR/VARCHAR function behavior
  • Incompatibility Function Code 2 Y9 SYSEM.VARCHAR(DECIMAL_EXPR)
  • DB2 V11 New Function Mode
    • VARCHAR (decimal column) CAST(decimal as VARCHAR) does not return
      • leading zeroes
      • Trailing decimal point character
      • Leading blanks for positive decimal values

<table>
<thead>
<tr>
<th></th>
<th>VARCHAR(000.1)</th>
<th>VARCHAR(1000.)</th>
<th>VARCHAR(1.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 11 NFM</td>
<td>'1'</td>
<td>'1000'</td>
<td>'1.1'</td>
</tr>
<tr>
<td>DB2 11 CM</td>
<td>'000.1'</td>
<td>'1000.'</td>
<td>'1.1'</td>
</tr>
</tbody>
</table>
Testing and Implementation

- We informed them of other functions (as VARCHAR_FORMAT) that would provide the same result.

- Application teams are able to test New Function Mode and go back using SET PATH statements.
  - This proved to be very helpful to show application teams how the results of CHAR and VARCHAR were different in SYSCOMPAT_V9 and CURRENT mode:
  - SET PATH statements:
    - SET PATH =syscompat_v9,SYSFUN,SYSPROC,SYSIBMADM,SYSBM;
    - SET PATH =syscurrent,SYSFUN,SYSPROC,SYSIBMADM,SYSBM;
Testing and Implementation: Fence IDs

• As a DB2 environment is set to CURRENT mode, it may be necessary to ‘fence’ IDs to provide ample time for application/SQL changes to be tested and implemented.

• Fence IDs in Profile tables of (non-batch) authorization ids

• SQLs containing CAST statements will not be ‘fenced’ even though authorization id is ‘fenced’

• Once application/SQLs are tested, delete rows in the profile tables.
Fence ID SQLs

- Fence ID SQLs:
  - Insert into `sysibm.dsn_profile(authid, profileid)` values ('AUTHID', 5001);
    - 5001 is a random number
  - Insert into `sysibm.dsn_profile_attributes(profileid, keywords, attribute1)` values (5001,'SPECIAL_REGISTER','SET PATH = "SYSCOMPAT_V9","SYSIBM","SYSFUN","SYSPROC","SYSIBMA DM"')
Be Aware…

- IFCID 366 vs IFCID 376 (Details vs Summary)
- Make sure trace is started automatically when DB2 starts
- Ensure that DB2 environments at the same maintenance levels and same module versions, otherwise I366 may not catch all incompatibilities
- I366 only reports on executed SQLs that return results
- Depending of level activity, you may find that old statements are flushed out of the STMTCACHE
- SQL TEXT sometimes not there "stmt id = 0"
  - Get the latest PI71903: ENHANCEMENTS FOR IFCID 366 AND IFCID 376
  (http://www-01.ibm.com/support/docview.wss?uid=swg1Pi71903&myns=swgimgmt&mynp=OCSSEPEK&mync=R&cm_sp=swgimgmt--OCSSEPEK--R)
- “Depends on Data”
Questions

• Contact Information:
  • Lillian Russell: lillian.russell@53.com
  • Lori Alexander: lori.alexander@53.com
  • Billy Sundarajnan: billy.sundarragan@53.com