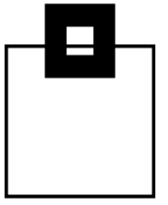
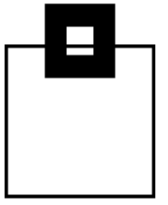


Mitigating Migration Mayhem in a Deprecated World

Roy Boxwell, SOFTWARE ENGINEERING GmbH

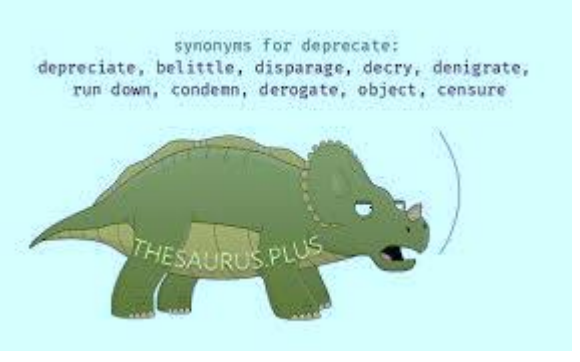
Agenda



- What, exactly, does “deprecated” mean?
- Why should I care?
- How do I find them all?
- How do I “fix” them all?
- Questions and Answers

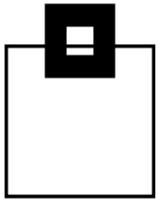
Agenda

- What, exactly, does “deprecated” mean?



- Why should I care?
- How do I find them all?
- How do I “fix” them all?
- Questions and Answers

What, exactly, does “deprecated” mean?



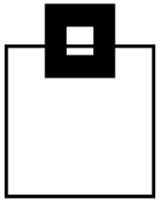
Deprecate:

1. Express disapproval of.
   “What I deprecate is persistent indulgence”
2. Another term for depreciate (sense 2).
   “He deprecates the value of children’s television”

Depreciate:

1. Diminish in value over a period of time
2. Disparage or belittle (something)

What, exactly, does “deprecated” mean?



So you can see that, in the IBM world, they have actually created a hybrid meaning of deprecate which is another term for depreciate but sense 1.

You could argue that this is typical behavior for IBM as they also mangle plurals all the time. My favorite is SYSIBM.SYSINDEXES – Still annoys me after 30 years although INDICES is actually more the Latin plural…

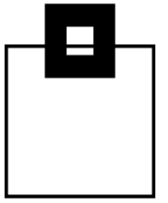
Anyway, long story short, deprecated for IBM means an item, object, code etc. that is no longer going to be updated and will, at some unknown point in the future, possibly, just might, disappear.

Agenda

- What, exactly, does “deprecated” mean?
- Why should I care?

- How do I find them all?
- How do I “fix” them all?
- Questions and Answers

Why should I care?



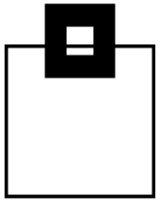
Good question!

The problem with all these dead parrots happily nailed to their perches is, right now, at this moment, they are probably not causing any problem.

At some point that will, like the parrots starting to smell, change…

For example: You use a SYNONYM? They have been deprecated for years but they still work – One day IBM Db2 will remove them and, because no one wants to change a running system, any of your programs that still use them will stop working… This is sub optimal…

Why should I care?



Another good reason is to make your Db2 catalog “modern”

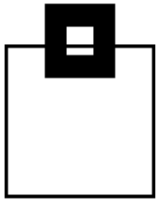
- Do you have any LARGE defined spaces?
- Do you have any zero DSSIZE objects?
- Do you have any classic index-based partitioning?
- Do you have any classic table-based partitioning?

All of these things will not *stop* your machine but they will cause software, in-house, 3rd party vendor, and also IBM to possibly hiccup at uncomfortable moments! Normally at 03:00 am on Sunday…

Agenda

- What, exactly, does “deprecated” mean?
- Why should I care?
- How do I find them all?
- How do I “fix” them all?
- Questions and Answers

How do I find them all?



Now we start looking into the depths of Db2 internals!

Starting with Db2 10, IBM introduced a new chapter in the "What's New?" book called "Deprecated function in Db2 nn" which contains this brief statement:

> Certain capabilities that Db2 nn for z/OS supports are *deprecated*, meaning that their use is discouraged. Although they are supported in Db2 nn, support is likely to be removed eventually.
>
> Avoid creating new dependencies that rely on deprecated function, and develop plans to remove any dependencies on such function.

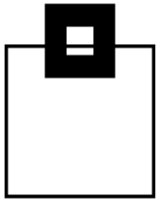
How do I find them all?

Did you all develop plans?

I bet you did! However, the chapter is actually a very good starting point for looking for deprecated items. For example the very first item in the long, long list (seven pages!) is:

Deprecated function	Recommended alternative	Support removed
6-byte RBA and LRSN format for the BSDS	Starting in DB2 11, convert the BSDS to use the extended 10-byte RBA and LRSN formats. The BSDS conversion must be completed before migration to DB2 12.	Db2 12

How do I find them all?



Thankfully, most of the other pages are ZPARM subsystem parameters that are changed during migration anyway or are never used and so have been simply removed.

But, apart from the six byte RBA, there were three others that are important:

Deprecated function	Recommended alternative	Support removed
BRF	Migrate to RRF	Db2 12
SIMPLE TS	Migrate to PBG, SEGM, PBR	Db2 12 FL504
SYNONYMS	Migrate to ALIAS	Db2 12 FL504 for create

How do I find them all?

Then along came Db2 11 which actually only added three things to the list of deprecated items:

Deprecated function	Recommended alternative	Support removed
HASH Tables	Drop hash	Db2 12 FL504
Non-UTS base tablespace	Migrate to PBG or PBR	Db2 12 FL504
SQL External Proc.	SQL Native	?

How do I find them all?

Then came Db2 12 which actually only added one thing to the list of deprecated items:

Deprecated function	Recommended alternative	Support removed
UNICODE column	ALTER to real UNICODE	Db2 12 FL500

These were created before Db2 12 FL500 by using:

colname VARCHAR(nn) CCSID 1208

colname VARGRAPHIC(nn) CCSID 1200

Style DDL syntax.

Under the covers Db2 created both of these as VARBIN but it also doubled the internal length of the VARGRAPHIC column…

How do I find them all?

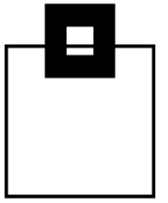
Why stop there? There a few other bits of data that I find very interesting from my Db2 catalog:

- **Empty databases**
- **Empty implicit databases**
- **Empty tablespaces**
- **Multi-table tablespaces**
- **How many tables in these multi-table tablespaces (DSMAX!)**
- **DSSIZE 0 objects**

So now, armed with a few SQLs, you can trawl through the Db2 Catalog and get a list of all the bad guys.

Following is just a list of all the queries you can use.

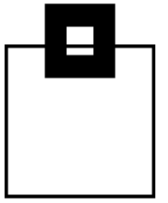
How do I find them all?



Empty databases:

```
SELECT NAME
FROM SYSIBM.SYSDATABASE DB
WHERE NOT EXISTS (SELECT 1
                  FROM SYSIBM.SYSTABLESPACE TS
                  WHERE DB.NAME = TS.DBNAME)
  AND NOT DB.NAME = 'DSNDB04'
  AND NOT DB.NAME = 'DSNDB01'
  AND NOT DB.NAME = 'DSNDB06'
  AND NOT DB.TYPE = 'W'
ORDER BY 1
FOR FETCH ONLY
WITH UR ;
```

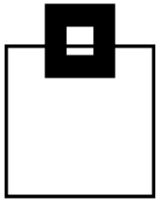
How do I find them all?



Count of empty implicit databases:

```
SELECT COUNT(*)
FROM SYSIBM.SYSDATABASE DB
WHERE     DB.IMPLICIT = 'Y'
  AND NOT DB.NAME     = 'DSNDB01'
  AND NOT DB.NAME     = 'DSNDB06'
  AND NOT EXISTS (SELECT 1
                  FROM SYSIBM.SYSTABLESPACE TS
                  WHERE DB.NAME = TS.DBNAME)
WITH UR ;
```

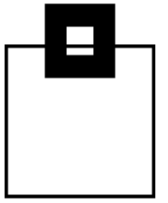
How do I find them all?



Empty tablespaces:

```
SELECT TS.DBNAME, TS.NAME
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSDATABASE   DB
WHERE     TS.NTABLES = 0
  AND     TS.DBNAME  = DB.NAME
  AND     DB.TYPE    = ' '
  AND NOT DB.NAME    = 'DSNDB01'
  AND NOT DB.NAME    = 'DSNDB06'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

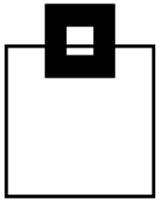
How do I find them all?



Hash-organized tablespaces:

```
SELECT TS.DBNAME, TS.NAME
      ,STRIP(TB.CREATOR) CONCAT '.' CONCAT STRIP(TB.NAME)
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSDATABASE   DB
    ,SYSIBM.SYSTABLES     TB
WHERE     TS.DBNAME            = DB.NAME
  AND     DB.TYPE              = ' '
  AND     TS.ORGANIZATIONTYPE = 'H'
  AND     TS.DBNAME            = TB.DBNAME
  AND     TS.NAME              = TB.TSNAME
  AND NOT DB.NAME              = 'DSNDB01'
  AND NOT DB.NAME              = 'DSNDB06'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

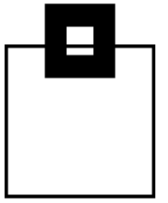
How do I find them all?



Zero size DSSIZE or Large defined partitioned tablespaces:

```
SELECT SUBSTR(TS.DBNAME , 1 , 8) AS DBNAME
     , SUBSTR(TS.NAME , 1 , 8)   AS NAME
     , TS.PARTITIONS
     , TS.DSSIZE
     , TS.TYPE
     , TS.SEGSIZE
FROM SYSIBM.SYSTABLESPACE TS
WHERE TS.PARTITIONS > 0
  AND (TS.DSSIZE    = 0
   OR  TS.TYPE      = 'L')
  AND NOT TS.DBNAME = 'DSNDB01'
  AND NOT TS.DBNAME = 'DSNDB06'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

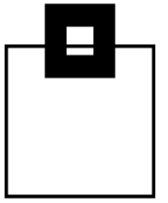
How do I find them all?



Classic index-based partitioning tables:

```
SELECT TS.DBNAME, TS.NAME
     , STRIP(TP.IXCREATOR) CONCAT '.' CONCAT STRIP(TP.IXNAME)
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSTABLEPART  TP
    ,SYSIBM.SYSDATABASE   DB
WHERE     TS.DBNAME     = DB.NAME
  AND     DB.TYPE       = ' '
  AND     TS.SEGSIZE    = 0
  AND     TS.PARTITIONS > 0
  AND     TS.TYPE      IN (' ' , 'L')
  AND     TS.DBNAME     = TP.DBNAME
  AND     TS.NAME       = TP.TSNAME
  AND NOT TP.IXCREATOR  = ''
  AND     TP.PARTITION  = 1
  AND NOT DB.NAME = 'DSNDB01'
  AND NOT DB.NAME = 'DSNDB06'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

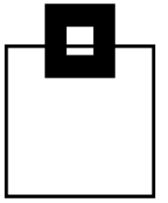
How do I find them all?



Classic table-based partitioning tables:

```
SELECT TS.DBNAME
     , TS.NAME
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSTABLEPART  TP
    ,SYSIBM.SYSDATABASE   DB
WHERE     TS.DBNAME     = DB.NAME
  AND     DB.TYPE       = ' '
  AND     TS.SEGSIZE    = 0
  AND     TS.PARTITIONS > 0
  AND     TS.TYPE      IN (' ' , 'L')
  AND     TS.DBNAME     = TP.DBNAME
  AND     TS.NAME       = TP.TSNAME
  AND     TP.IXCREATOR  = ''
  AND     TP.PARTITION  = 1
  AND NOT DB.NAME = 'DSNDB01'
  AND NOT DB.NAME = 'DSNDB06'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

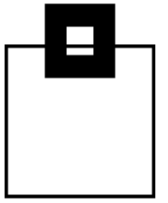
How do I find them all?



Segmented or simple tablespaces with one table:


```
SELECT TS.DBNAME
     , TS.NAME
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSDATABASE   DB
WHERE     TS.DBNAME     = DB.NAME
  AND     DB.TYPE       = ' '
  AND NOT DB.NAME       = 'DSNDB01'
  AND NOT DB.NAME       = 'DSNDB06'
  AND     TS.PARTITIONS = 0
  AND     TS.TYPE       = ' '
  AND     TS.NTABLES    = 1
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

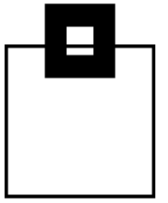
How do I find them all?



How many tables in multi-table tablespaces:

```
SELECT TS.DBNAME
     , TS.NAME
     , SUM(TS.NTABLES)
FROM SYSIBM.SYSTABLESPACE TS
WHERE     TS.NTABLES > 1
  AND NOT TS.DBNAME  = 'DSNDB01'
  AND NOT TS.DBNAME  = 'DSNDB06'
GROUP BY TS.DBNAME, TS.NAME
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```

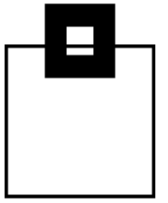
How do I find them all?



BRF table partitions:

```
SELECT TS.DBNAME
     , TS.NAME
     , TP.PARTITION
FROM SYSIBM.SYSTABLESPACE TS
    ,SYSIBM.SYSTABLEPART  TP
WHERE NOT TS.DBNAME = 'DSNDB01'
  AND NOT TS.DBNAME = 'DSNDB06'
  AND     TS.DBNAME = TP.DBNAME
  AND     TS.NAME   = TP.TSNAME
  AND NOT TS.TYPE   = 'O'
  AND     TP.FORMAT = ' '
ORDER BY 1 , 2 , 3
FOR FETCH ONLY
WITH UR ;
```

How do I find them all?

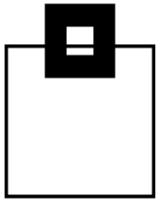


Six byte RBA table partitions:

```
SELECT TP.DBNAME
     , TP.TSNAME
     , TP.PARTITION
     , TP.RBA_FORMAT
FROM SYSIBM.SYSTABLEPART TP
    ,SYSIBM.SYSDATABASE  DB
WHERE     TP.DBNAME     = DB.NAME
  AND     DB.TYPE       = ' '
  AND NOT DB.NAME       = 'DSNDB01'
  AND NOT DB.NAME       = 'DSNDB06'
  AND    (TP.RBA_FORMAT = ' '
       OR TP.RBA_FORMAT = 'B')
ORDER BY 1 , 2 , 3
FOR FETCH ONLY
WITH UR  ;
```


Partition is 0 for TS objects.

How do I find them all?

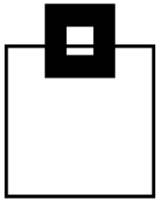


Six byte RBA index partitions:

```
SELECT STRIP(IP.IXCREATOR) CONCAT '.' CONCAT STRIP(IP.IXNAME)
     , IP.PARTITION
     , IP.RBA_FORMAT
FROM SYSIBM.SYSINDEXPART IP
    ,SYSIBM.SYSINDEXES   IX
WHERE     IX.CREATOR    = IP.IXCREATOR
  AND     IX.NAME       = IP.IXNAME
  AND NOT IX.DBID      IN (1 , 6)
  AND    (IP.RBA_FORMAT = ' '
       OR IP.RBA_FORMAT = 'B')
ORDER BY 1 , 2 , 3
FOR FETCH ONLY
WITH UR   ;
```


Partition is 0 for IX objects.

How do I find them all?



Non-SMS VOLUME usage:

```
SELECT STRIP(VO.SGNAME)
      ,STRIP(VO.VOLID)
FROM SYSIBM.SYSVOLUMES VO
WHERE NOT VO.VOLID = '*'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```


Not actually deprecated but why would you be using specific VOLIDs these days?

The use of VOLUMES in CREATE STOGROUP was made optional way back in DB2 V9

How do I find them all?

Synonyms:

```
SELECT STRIP(SY.CREATOR)    CONCAT '.' CONCAT STRIP(SY.NAME)
     , STRIP(SY.TBCREATOR) CONCAT '.' CONCAT STRIP(SY.TBNAME)
FROM SYSIBM.SYSSYNONYMS SY
ORDER BY 1
FOR FETCH ONLY
WITH UR  ;
```

How do I find them all?

Unicode columns:

```
SELECT STRIP(CO.TBCREATOR) CONCAT '.' CONCAT STRIP(CO.TBNAME)
     , STRIP(CO.NAME)
     , CO.LENGTH
     , CO.CCSID
FROM SYSIBM.SYSCOLUMNS CO
WHERE CO.CCSID  IN ( 1200 , 1208 )
  AND CO.COLTYPE = 'VARBIN'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR  ;
```

How do I find them all?

SQL External Procedures:

```
SELECT STRIP(RO.SCHEMA) CONCAT '.' CONCAT
       STRIP(RO.NAME)
      ,STRIP(RO.SPECIFICNAME)
FROM SYSIBM.SYSROUTINES RO
WHERE RO.ROUTINETYPE   = 'P'
  AND RO.ORIGIN        = 'E'
  AND RO.FUNCTION_TYPE = ' '
  AND RO.LANGUAGE      = 'SQL     '
FOR FETCH ONLY
WITH UR  ;
```

How do I find them all?

Why were the DSNDB01 (DBID = 1) and DSNDB06 (DBID = 6) excluded in all of the queries you may well ask?

Because the Db2 Directory and Catalog still contains:

- **Simple spaces**
- **Multi-table spaces**
- **Six byte RBA DEFINE NO table partitions**

Now it could be a coincidence but only eight weeks after the freeware software that this presentation is based on (www.segus.com) was first delivered to test customers, IBM developed a fix for some of these deprecated items! Check out:

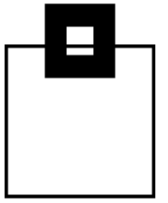
PH31798: ADD NEW DB2 12 TABLE SPACES TO THE DSNTIJCV JOB

How do I find them all?

As just mentioned, SOFTWARE ENGINEERING/SEGUS brought out a freeware product called Migration HealthCheck in the PocketTools range.

Just visit the web site, register to access the download site and send us your CPU model/make (output from a /D M=CPU command in SDSF) and we will send you a password valid for a year. There is also a buyable version that then generates all of the ALTERS and SQL DDL to correct all of the found problems.

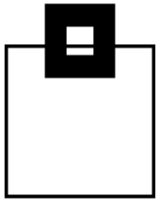
How do I find them all?



When you run it the output looks like (1 of 5):

```
Db2 Migration HealthCheck V1.5 for QB1A DSN11015  started at 2021-03-11-14.59.21
                    Lines with *** are deprecated features

Number of DATABASES             :       294
  # of empty DATABASES          :       105
  # of implicit DATABASES       :       183
  # of empty implicit DATABASES:       102

Number of TABLESPACES           :      3681
  of which HASH organized       :         0
  of which PARTITIONED CLASSIC  :        10 ***
    # Partitions                :      1218 ***
  of which SEGMENTED            :      1727 ***
  of which SIMPLE               :         0
  of which LOB                  :        53
  of which UTS PBG              :      1861
    # Partitions                :      1863
  of which UTS PBR              :        10
    # Partitions                :       405
  of which XML                  :        20
```

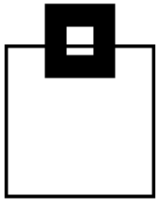
How do I find them all?



When you run it the output looks like (2 of 5):

```
Number of tablespaces as LARGE :         4 ***
Number of empty tablespaces    :         9
Number of BRF table partitions :         2 ***
Number of multi-table TSs      :        31
  # of tables within these     :       143

Number of tables               :      7522
  of which ACCELERATOR ONLY    :         0
  of which ALIASes             :      3651
  of which ARCHIVEs            :         0
  of which AUXs                :        46
  of which CLONEs              :         2
  of which GTTs                :       106
  of which HISTORYs            :         1
  of which MQTs                :         1
  of which TABLEs              :      3674
  of which VIEWs               :        21
  of which XMLs                :        20

Number of SYNONYMs             :         2 ***
```

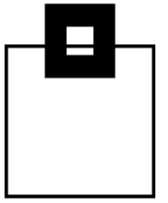
How do I find them all?



When you run it the output looks like (3 of 5):

```
Number of UNICODE V11 Columns  :          2 ***

Number of PROCEDURES           :        110
  of which SQL EXTERNAL        :          1 ***
  of which EXTERNAL            :        108
  of which NATIVE SQL          :          1

Number of FUNCTIONS            :         90
  of which EXTERNAL TABLE      :         38
  of which EXTERNAL SCALAR     :         42
  of which SOURCED AGGREGATE   :          0
  of which SOURCED SCALAR      :          0
  of which SQL TABLE           :          1
  of which SQL SCALAR          :          9
  of which SYSTEM-GENERATED    :          0
```

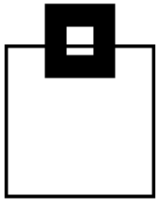
How do I find them all?



When you run it the output looks like (4 of 5):

```
Number of Indexes              :      4289
  of which HASH                :         0
  of which type 2              :      4265
    # of partitioned IXs       :         5 ***
    # Partitions               :       155
  of which DPSI                :        13
    # Partitions               :       513
  of which PI                  :        11
    # Partitions               :      1426

Number of table partitions     :      5286
  of which DEFINE NO           :      1665
  of which six byte RBA <11 NFM:         0
    of which DEFINE NO         :         0
  of which six byte RBA Basic  :       131 ***
    of which DEFINE NO         :         0
  of which ten byte RBA        :      3490
    of which DEFINE NO         :         0
  of which unknown RBA         :      1665
    of which *not* DEFINE NO   :         0
```

How do I find them all?



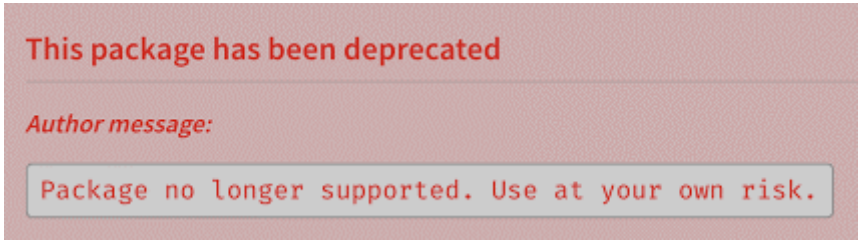
When you run it the output looks like (5 of 5):

```
Number of index partitions     :      6354
  of which DEFINE NO           :      1906
  of which six byte RBA <11 NFM:         0
    of which DEFINE NO         :         0
  of which six byte RBA Basic  :         1 ***
    of which DEFINE NO         :         0
  of which ten byte RBA        :      4447
    of which DEFINE NO         :         0
  of which unknown RBA         :      1906
    of which *not* DEFINE NO   :         0

Number of STOGROUPS            :         8
Number of non-SMS VOLUMES      :         0

Number of PLANs                :        53
Number of PACKAGES (total)     :      5583
Number of PACKAGES (distinct)  :       558
Number of SQL statements       :    408265

Db2 Migration HealthCheck V1.5 for QB1A DSN11015  ended at 2021-03-11-14.59.24

Db2 Migration HealthCheck ended with RC:    0
```

How do I find them all?

It also outputs a list of everything found:

```
Empty DB: DSN00235
Empty DB: SAXDBP2
Segmented DB: ROYXTEST TS: DBRMTS01
Classic partitioned DB: DOGTEST1 TS: DOGS02
Synonym: BOXWEL2.EMILYB
 for SYSIBM.SYSTABLES
  Has the following SYSIBM.SYSPACKDEP:
  DCOLLID   : MDB2VNEX_TEST
  DNAME     : MORE0001
  DCONTOKEN : 1B1E7E6F0E8FEFE0
  DTYPE     : Not a Trigger or native SQL package
UNICODE column defined in table DB2V11.DB2V11TB1
  Column X06
STOGROUP.VOLID Non-SMS: ROYBOY1.GEORGE
```



How do I find them all?

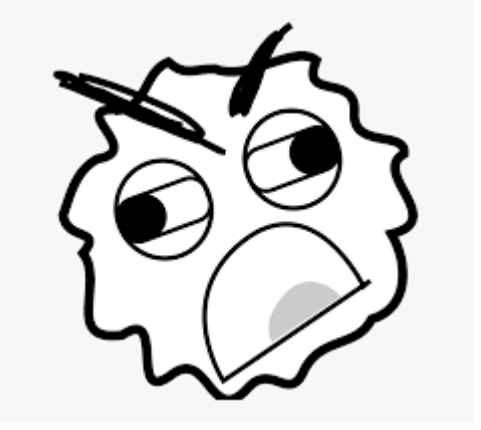
If the optional paid version is licensed a list of ALTERs and DROPs:

```
DROP DATABASE DSN00235 ;
COMMIT ;
DROP DATABASE SAXDBP2 ;
COMMIT ;
ALTER TABLESPACE ROYXTEST.DBRMTS01 MAXPARTITIONS 1 ;
COMMIT ;
ALTER INDEX DOGTEST1.INDEX_GREATER_THAN_EIGHT_ON_DOGTAB2
  NOT CLUSTER ;
ALTER INDEX DOGTEST1.INDEX_GREATER_THAN_EIGHT_ON_DOGTAB2
      CLUSTER ;
COMMIT ;
ALTER TABLESPACE DOGTEST1.DOGS02 SEGSIZE 64 ;
COMMIT ;
SET CURRENT SQLID = 'BOXWEL2' ;
DROP SYNONYM EMILYB ;
COMMIT ;
CREATE ALIAS BOXWEL2.EMILYB
FOR SYSIBM.SYSTABLES ;
COMMIT ;
ALTER TABLE DB2V11.DB2V11TB1
  ALTER COLUMN X06
    SET DATA TYPE VARCHAR(   25)
;
COMMIT ;
ALTER STOGROUP ROYBOY1
  REMOVE VOLUMES ('GEORGE') ;
COMMIT ;
```

Agenda

- What, exactly, does “deprecated” mean?
- Why should I care?
- How do I find them all?
- How do I “fix” them all?
- Questions and Answers

How do I fix them all?



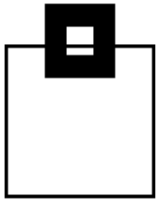
Caveat Emptor!

Doing these fixes will "fix" the deprecated items but it will invalidate any and all of your packages that refer to them!

If objects are dropped then any GRANTS will also be lost!

Remember to gather all GRANTs before you begin!

How do I fix them all?



Empty databases:

Fix: `DROP DATABASE xxxxxxxx ;`

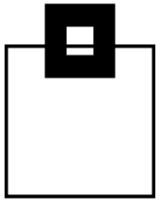
Empty tablespaces:

Fix: `DROP TABLESPACE xxxxxxxx.yyyyyyyy ;`

Hash-organized tablespaces:

Fix: `ALTER TABLE xxxxxxxx.yyyyyyyy DROP ORGANIZATION ;`
Then possibly create a new index for normal access as the Hash Index is automatically dropped by using this command.

How do I fix them all?



Classic index-based partitioning tables:

Two-stage fix:
```
ALTER INDEX aaa.bbb NOT CLUSTER ;
ALTER INDEX aaa.bbb CLUSTER ;
```
Now you have table-based so now just the flip to UTS PBR:
```
ALTER TABLESPACE ddd.eee SEGSIZE 64 ;
```

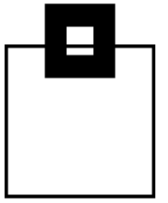

Classic table-based partitioning tables or zero DSSIZE:

Fix:
```
ALTER TABLESPACE ddd.eee SEGSIZE 64 ;
```


Segmented or simple tablespaces with single tables:

Fix:
```
ALTER TABLESPACE ddd.eee MAXPARTITIONS 1 ;
```
For multi-table tablespaces you must go to Db2 12 FL508.

How do I fix them all?



The never ending saga of PBR Conversion…

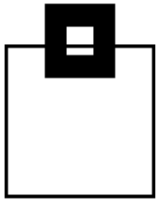
We all know that the transition from PBR UTS to PBR RPN UTS is painful…
You must do a TS level reorg but with COPY TEMPLATEs at the TP level
Most peoples PBRs are pretty big and if you have 4096 partitions and attempt to put these to TAPE you will have trouble!

In January 2017 a problem was opened:

PI75518: REORG PARTLEVEL WITH INLINE IMAGE COPY ON TAPE USES TOO MANY TAPE DRIVES

As of March 2021 it is still OPEN…

How do I fix them all?



BRF table partitions:

Fix: `REORG`

Six byte RBA index or table partitions:

Fix: `REORG`

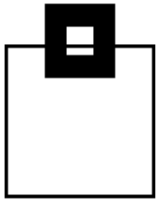
For DEFINE NO objects only a DROP and a CREATE will “fix” the problem – However watch out for any dependencies as even DEFINE NO can be referred to by SQL of course!

STOGROUP with non-SMS VOLID:

Fix:

```
ALTER STOGROUP xxx REMOVE VOLUMES ('yyy') ;
```

How do I fix them all?



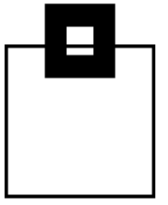
Deprecated work files?

As you have seen workfiles are deliberately excluded from this discussion as their usage is a bit “unclear”…

- **If your workfile tablespace is segmented and non-UTS and with zero as a SECQTY it will be used for CTTs, Large Sorts, Materializing Views etc. which can span more than one tablespace**
- **If your workfile tablespace is a PBG with or without a SECQTY it will be used for DGTTs, Scrollable cursors and SQL MERGE operations where the data cannot span more than one tablespace**

You normally need both!
Then a few ZPARMs rear their ugly heads…

How do I fix them all?

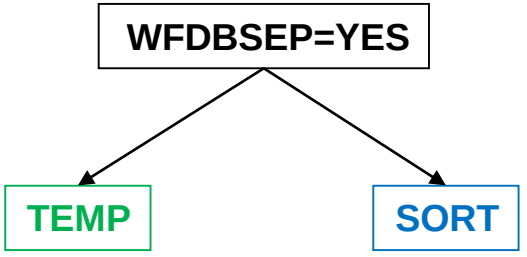


WFDBSEP default is NO. If set to YES Db2 will, for DGTT usage etc., allocate to PBG or Segmented with non zero secqty. If non-workfile usage is required it will attempt to allocate to segmented non-UTS with zero SECQTY. If either of these selections fails -904 is returned.
If set to NO it will still attempt the “preferred space” but if none are available it will fail over to another type of workfile tablespace.

MAXTEMPS default is zero. You can put in here a number of MB or GB that is the limit an agent can allocate. This is quite handy for stopping run-away cartesian join style transactions.

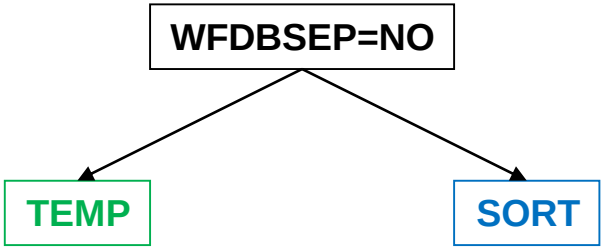
WFSTGUSE_AGENT_THRESHOLD default is zero. It can send an alert when nn% of all workfiles are in use by a single agent. You could set this to, say, 30 and monitor the xxxxMSTR to see who is hogging the workfile space and take corrective actions.

Selection order: PBG and WFDBSEP



- 32K PBG
- 4K PBG
- 32K segm
  - SECQTY>0
- 4K segm
  - SECQTY>0

- 32K segm
  - SECQTY=0
- 4K segm
  - SECQTY=0



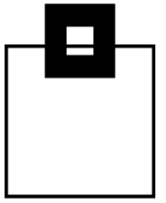
- 32K PBG
- 4K PBG
- 32K segm
  - SECQTY>0
- 4K segm
  - SECQTY>0
- 32K segm
  - SECQTY=0
- 4K segm
  - SECQTY=0

- 32K segm
  - SECQTY=0
- 4K segm
  - SECQTY=0
- 32K segm
  - SECQTY>0
- 4K segm
  - SECQTY>0
- 32K PBG
- 4K PBG

PBGs are selected independent of their SECQTY

Thanks to Peter Hartmann for the use of this graphic

How do I fix them all?



UNICODE columns:

For CCSID 1208 you do this fix:

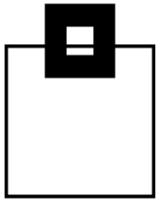
```
ALTER TABLE aaa.bbb
   ALTER COLUMN ccc
      SET DATA TYPE VARCHAR(ll)
;
```

For CCSID 1200 you do this fix:

```
ALTER TABLE aaa.bbb
   ALTER COLUMN ccc
      SET DATA TYPE VARGRAPHIC(ll/2)
;
```

Where `ll` is the LENGTH from SYSIBM.SYSCOLUMNS.

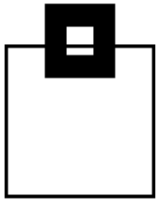
How do I fix them all?



SYNONYMS (page 1 of 6):
These are "tricky" and you must take a multi-modal approach. Start with dependency checks on the SYNONYM to actually see if it might cause problems by the DROP:


```
SELECT BCOLNAME
     , BOWNER
     , DSCHEMA
     , DNAME
     , DCOLNAME
     , CASE DTYPE
       WHEN 'B' THEN 'Basic Trigger           '
       WHEN 'C' THEN 'Generated Column        '
       WHEN 'F' THEN 'Function                '
       WHEN 'I' THEN 'Index                   '
       WHEN 'M' THEN 'Materialized Query table'
       WHEN 'O' THEN 'Procedure               '
```

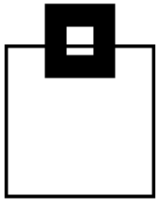
How do I fix them all?



SYNONYMS (page 2 of 6):

```
       WHEN 'O' THEN 'Procedure                '
       WHEN 'V' THEN 'View                     '
       WHEN 'X' THEN 'Row Permission           '
       WHEN 'Y' THEN 'Column Mask              '
       WHEN '1' THEN 'Advanced Trigger         '
       ELSE          'Unknown                  '
       END
     , DOWNER
FROM SYSIBM.SYSDEPENDENCIES
WHERE BSCHEMA = 'Synonym Schema'
  AND BNAME   = 'Synonym Name'
  AND BTYPE = 'S'
ORDER BY 1 , 2 , 3 , 5 , 6 , 7
FOR FETCH ONLY
WITH UR ;
```

How do I fix them all?

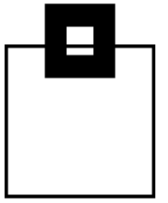


SYNONYMS (page 3 of 6):

The output shows you any extra work you might have to do! Like DROPping any dependent FUNCTIONs, VIEWs or MQTs etc. Then you run another SQL to check out any package dependencies:


```
SELECT DCOLLID
     , DNAME
     , HEX(DCONTOKEN)
     , CASE DTYPE
       WHEN 'F' THEN 'Compiled SQL scalar function       '
       WHEN 'N' THEN 'Native SQL routine package         '
       WHEN 'O' THEN 'Original copy of a package         '
       WHEN 'P' THEN 'Previous copy of a package         '
       WHEN 'R' THEN 'Reserved for IBM use               '
       WHEN 'T' THEN 'Basic Trigger                      '
```

How do I fix them all?

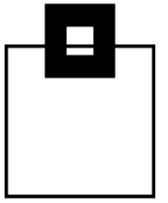


SYNONYMS (page 4 of 6):

```
       WHEN ' ' THEN 'Not a Trigger or native SQL package'
       WHEN '1' THEN 'Advanced Trigger                    '
       ELSE          'Unknown                             '
       END
     , DOWNER
FROM SYSIBM.SYSPACKDEP
WHERE BQUALIFIER = 'Synonym Schema'
  AND BNAME      = 'Synonym Name'
  AND BTYPE      = 'S'
ORDER BY 1 , 2
FOR FETCH ONLY
WITH UR ;
```


This output shows you a list of packages that will require at least a REBIND after your have got rid of the SYNONYM.

How do I fix them all?



SYNONYMS (page 5 of 6):

Then, finally, comes the fix:

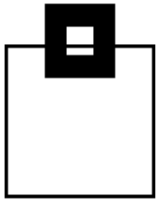
Generate a set of SPUFI DDL statements:

```
SET CURRENT SQLID = 'Synonym schema' ;
DROP SYNONYM 'Synonym name' ;
COMMIT ;
CREATE ALIAS 'Synonym schema'.'Synonym name'
       FOR   'Table creator'.'Table name' ;
COMMIT ;
```


GRANTs do not have to be checked for SYNONYMS as GRANTs are recorded in the catalog against the underlying TABLEs which have not been dropped.

Once this is done then recreate all of the dependent objects that had to be dropped and all of their GRANTs and any of their dependencies as well.

How do I fix them all?

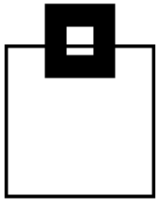


SYNONYMS (page 6 of 6):

So what is the difference between a SYNONYM and an ALIAS anyway?

Characteristic	Synonyms (deprecated)	Aliases
Can be created in application compatibility V12R1M504 and higher?	No	Yes
Requires authorization to create?	No	Yes
Can be defined on objects not at the current server?	No	Yes
Can be defined on the name of an object that does not yet exist?	No	Yes, but it must exist when used
Is dropped when referenced objects are dropped?	Yes	No
Uses a qualified object name for the object?	No, one-part name	Yes
Can be referenced or used by users other than the object owner?	No	Yes

How do I fix them all?



Procedure – External SQL (page 1 of 2):

First get/extract CREATE PROCEDURE and GRANT DDL

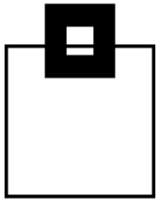
DROP Procedure

Change CREATE Procedure syntax by removing keywords:

- FENCED
- EXTERNAL

If the WLM ENVIRONMENT keyword is there either remove it as well or add FOR DEBUG MODE

How do I fix them all?



Procedure – External SQL (page 2 of 2):

Check the Procedure code for any use of unqualified names that refer to Columns, SQL Variables or Parameters:

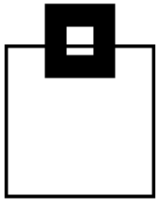
- In an EXTERNAL SQL Procedure Db2 first matches Variables or Parameters and then Columns
- In a NATIVE SQL Procedure Db2 first matches Columns and then Variables or Parameters

CREATE Procedure

GRANT permissions

Possibly adjust any TIME=nnn parameters in the JCL as EXTERNAL were charged to WLM whereas NATIVE is charged to the user

Agenda



- What, exactly, does “deprecated” mean?

- Why should I care?

- How do I find them all?

- How do I “fix” them all?
- Questions and Answers

Questions & Answers

ZOWE update!!

We have installed the 1.20 version.

Our WorkLoadExpert (WLX) tool, including all of its Audit sub-components, is now complete.

Our SpaceAssuranceExpert (SAX) tool is now in beta test.

Our SpaceManager (SDB2) tool is next in line!

Coffee cups!!

Just like Benedict: If you would like a free coffee cup just send an email (with your full postal address) to techsupport@seg.de and we will send it out!