

# Db2 for z/OS Know your Limits!

## *SOFTWARE ENGINEERING*

Session code: V09

Wednesday October 23<sup>rd</sup> 2019 – 10:45-11:45

Db2 z/OS

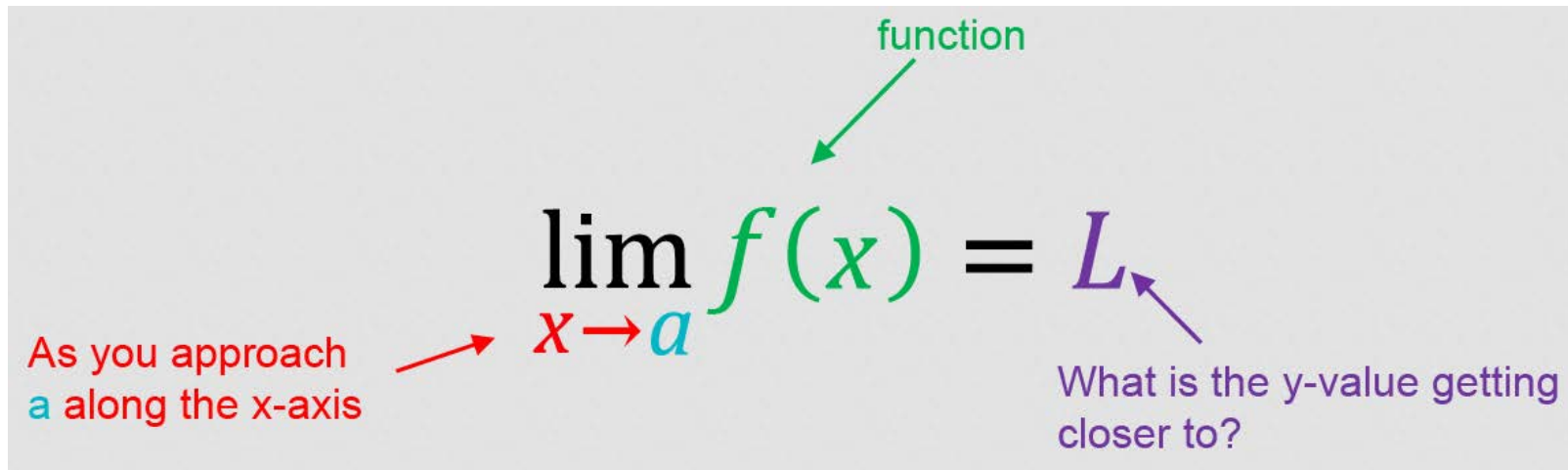


**IDUG**

Leading the Db2 User  
Community since 1988



# Db2 for z/OS Know your Limits!



The diagram shows the mathematical expression  $\lim_{x \rightarrow a} f(x) = L$  with several annotations:

- A green arrow points from the word "function" to the  $f(x)$  part of the equation.
- A red arrow points from the text "As you approach a along the x-axis" to the  $x \rightarrow a$  part of the equation.
- A purple arrow points from the text "What is the y-value getting closer to?" to the  $L$  part of the equation.

# Agenda

- Limits through the machine
- Limits through logic
- Limits through design
- Future directions



# Agenda

- Limits through the machine
- Limits through logic
- Limits through design
- Future directions



# Limits through the machine

They say that space is infinite - up to a point

The same is true for Db2!

If you can imagine having infinite disk space and infinite memory,  
what limits could there possibly be?



# Limits through the machine

In the beginning was the VSAM Cluster...

Well before Db2 saw the light of day  
Older than me!

Size limit of 4GB per dataset - that is HUGE!  
We will never get that much data...

Remember Bill Gates and the apocryphal story about 640KB of RAM...

# Limits through the machine

The VSAM limit is actually split into different limits that the DBA and Db2 must work with:

- Simple/Segmented VSAM Linear Dataset (LDS from now on) is limited to 2GB
- Partitioned Space 1GB, 2GB, 4GB, 8GB, 16GB, 32GB, 64GB
  - For larger than 4GB objects you must have a data class in SMS with the Extended Format and Extended Addressability set
- Non-partitioned indexes also have PIECESIZE which starts at 256KB and then binary steps all the way up to 268435456KB !
  - Thankfully you can also allocate in MB or GB

# Limits through the machine

Piecesize continued...

However any value larger than 2GB requires DSSIZE or LARGE on the tablespace definition

Here you can also see that a seemingly simple question:

How many pieces (datasets) can my index have?

Gets a rather complex answer:

$$\text{MIN} ( 4096 , 2^{32} / ( \text{DSSIZE} / \text{TS PGSIZE} ) )$$

Eg: 128GB DSSIZE with 8KB Tablespace Page Size gives 256 Pieces (datasets)  
or for a 4GB DSSIZE with 4KB Tablespace Page Size gives 4096 Pieces (datasets).

All clear on that ??



# Limits through the machine

Hooray for Db2 12 FL500!

In Db2 12 we get the Relative Page Numbering (RPN) PBR with variable DSSIZE and *\*also\** a variable DSSIZE for the partitioning indexes. Plus the DSSIZE GB does not have to be a binary number.

To do this the RID has increased to seven bytes but it completely decouples the number of partitions from the equation - which is a very good thing!

However, the rule is still true that for larger than 4GB objects you must have a data class in SMS with the Extended Format and Extended Addressability set.

# Limits through the machine

These dataset limits then go into the LDS arena:

- A simple or segmented space can have 32 LDSs
- A non-partitioning index can have a “number” of pieces
- A LOB space can have 254 LDSs
- A Partition can only have one of course!

# Limits through the machine

This is then the first set of limits for us.

You must monitor how many LDSs all of your simple, segmented and LOB objects have and you must get warned well before you hit the buffers! If you have 32 LDSs the REORG might take a while...

Partitioned objects are of course different...

Here you must monitor how full each partition is and how full each index is and how many pieces there are. Is that all?

# Limits through the machine

No! Of course not...

The Partition By Growth (PBG) Universal Table Space (UTS) construct brought with it some “new” problems:

- 1) No partitioning index allowed – only NPSIs on these
- 2) `MAXPARTITIONS` is the new LDS limit

So, for all your PBGs you must monitor not only how many partitions they currently have, but also how full is that very last partition when you have it allocated!

# Limits through the machine

**SpaceAssuranceExpert (SAX)** to the rescue!

It uses IFCIDs to get the allocation of extents and LDSs in real-time and to take action instantly to stop any -904's from happening!

Ok, so now you are checking how many LDSs of which size for all of your different objects – Everything must be ok now??

# Limits through the machine

No! Of course not...

Think SMS Copy Pool sizes...When you are using [flashcopy](#) or just normal [Image Copy](#) you must guarantee that you have enough space to actually do all the copies you want to do...

Normally this job is done by the storage team but I think it would be a good idea if the DBA also checked whether or not the copy pool is up to its name

or is it only a puddle?



# Limits through the machine

No! Of course not...

Think SMS Copy Pool sizes...When you are using **flashcopy** or just normal **Image Copy** you must guarantee that you have enough space to actually do all the copies you want to do...

Normally this job is done by the storage team but I think it would be a good idea if the DBA also checked whether or not the copy pool is up to its name

or is it only a puddle?

**SpaceAssuranceExpert (SAX)** to the rescue!

It can easily monitor *\*all\** of your SMS Storage groups!

# Agenda

- Limits through the machine
- Limits through logic
- Limits through design
- Future directions



# Limits through logic

Now we get to the part where Db2 constructs start saying “no”

Think SEQUENCES here... A quick look in the SQL guide will show you that when you create a SEQUENCE:

```
CREATE SEQUENCE ROY_TEST_SEQ_ASC
  START WITH 1
  INCREMENT BY 1
  MAXVALUE 9999
  CACHE 10;
```

# Limits through logic

You actually get:

```
CREATE SEQUENCE ROY_TEST_SEQ_ASC  
    START WITH 1  
    INCREMENT BY 1  
    MAXVALUE 9999  
    NO CYCLE  
    CACHE 10;
```

The “NO CYCLE” is the, by default, bad guy here...  
This is a -904 waiting to happen...

# Limits through logic

For all of SEQUENCES you must periodically see how close to the top or the bottom of your available range they are.

This is also true for Identity columns and XML DOC Ids:

```
SEQTYPE CHAR(1) NOT NULL
```

Type of sequence object:

- A** Alias for a sequence
- I** An identity column
- S** A user-defined sequence
- X** An implicitly created DOCID  
column for a base table that contains XML data.

You want to get these *\*way\** before they hit the buffers!

## Limits through logic

Now in the time before SEQUENCES lots of shops were using **SMALLINT**, **INTEGER** or **DECIMAL** defined fields in their tables' Primary Keys doing exactly what a modern sequence does.

Naturally no-one has “updated” these pre-sequence sequences to use proper sequences and so you have another layer of danger lurking out there...



# Limits through logic

You must find all your **numerically** defined **primary key** fields and see if they are approaching the Db2 numeric limits...

<b>Item</b>			<b>Limit</b>
Smallest	SMALLINT	value	-32768
Largest	SMALLINT	value	32767
Smallest	INTEGER	value	-2147483648
Largest	INTEGER	value	2147483647
Smallest	BIGINT	value	-9223372036854775808
Largest	BIGINT	value	9223372036854775807
Smallest	DECIMAL	value	$1 - 10^{31}$
Largest	DECIMAL	value	$10^{31} - 1$

# Limits through logic

But what are you checking?

Are you looking at just the Db2 Catalog values after a RUNSTATS?

Or

Are you selecting all the columns dynamically from the User Data so that you get real values?

# Limits through logic

But what are you checking?

Are you looking at just the Db2 Catalog values after a RUNSTATS?

Or

Are you selecting all the columns dynamically from the User Data so that you get real values?

Horses for courses - as we say in England...

**SpaceAssuranceExpert (SAX)** to the rescue!

All SEQUENCES, Doc Ids, IDENTITY Columns and Primary Key numeric columns are validated automatically. If desired directly from the User Data.

# Agenda

- Limits through the machine
- Limits through logic
- Limits through design
- Future directions



# Limits through design

Design limits that require checking:

- How many columns in a table?
- How many columns in an index?
- How many bytes long is my index?
- Is there a Database limit?
- Is there an Object limit?
- Death by DBAT?



**EVEN THE  
NICEST  
PEOPLE  
HAVE  
THEIR  
LIMITS.**

# Limits through design

How many columns in a table?

From day one Db2 has allowed **750 columns** as the absolute maximum. Depending on View definitions you can actually be forced to have less...

Something to check just in case that **ALTER ADD COLUMN** is gonna fail horribly...



# Limits through design

How many columns in a table?

From day one Db2 has allowed **750 columns** as the absolute maximum. Depending on View definitions you can actually be forced to have less...

Something to check just in case that **ALTER ADD COLUMN** is gonna fail horribly...

**SpaceAssuranceExpert** (SAX) to the rescue!  
SAX validates how near you are to the cliff edge...

# Limits through design

## How many columns in an index?

From day one Db2 has allowed **64 columns** as the absolute maximum. However the byte count varies depending on whether or not the index is a good old partitioning index (PI) or any other index.

For a PI you get a maximum size for:

- **PADDED indexes of  $255 - n - 3d$  bytes**
- **NOT PADDED indexes of  $255 - n - 2m - 3d$  bytes**

Where:

**n** is the number of columns which are NULLable

**m** is the number of varying length columns

**d** is the number of DECFLOAT columns

# Limits through design

For any other indexes you get a maximum size for :

- PADDED indexes of  $2000 - n - 3d$  bytes
- NOT PADDED indexes of  $2000 - n - 2m - 3d$  bytes

Where:

**n** is the number of columns which are NULLable

**m** is the number of varying length columns

**d** is the number of DECFLOAT columns

# Limits through design

For any other indexes you get a maximum size for :

- PADDED indexes of  $2000 - n - 3d$  bytes
- NOT PADDED indexes of  $2000 - n - 2m - 3d$  bytes

Where:

**n** is the number of columns which are NULLable

**m** is the number of varying length columns

**d** is the number of DECFLOAT columns

**SpaceAssuranceExpert (SAX)** to the rescue!

SAX validates how near you are to the buffers...

# Limits through design

How many bytes long is my index?

In Db2 12 IBM came up with the Fast Traversal Block (FTB) which most people call Fast Index Traversal (FIT) and this comes with a bunch of limits:

- 1) Must be a unique index
- 2) Must have a total length  $\leq 64$  bytes
- 3) No TIMEZONE usage
- 4) No active versioning
- 5) Index partitions must have less than 2,000,000 leaf pages

## Limits through design

So you can see that you might well be using a FIT and then do e.g.

```
ALTER INDEX x.y ADD INCLUDE COLUMN ( col1 )
```

and **\*boom\*** the index is no longer eligible for FIT usage...

Check before **\*every\*** ALTER INDEX whether or not the index is used by FIT and whether or not your ALTER will tip it over the edge!

## Limits through design

So you can see that you might well be using a FIT and then do e.g.

```
ALTER INDEX x.y ADD INCLUDE COLUMN ( col1 )
```

and **\*boom\*** the index is no longer eligible for FIT usage...

Check before **\*every\*** ALTER INDEX whether or not the index is used by FIT and whether or not your ALTER will tip it over the edge!

**SpaceAssuranceExpert** (SAX) to the rescue!

SAX validates how near you are to the tipping point...



# Limits through design

Is there a Database limit?

Yes indeed there is, but nowadays at **65,217** (Back in DB2 V5.1 it was 32,511) it is pretty hard to reach!

However, if you have a lot of implicit Tablespaces where no-one tidies up and DROPs the implicit database it can add up quickly!

# Limits through design

Is there a Database limit?

Yes indeed there is, but nowadays at **65,217** (Back in DB2 V5.1 it was 32,511) it is pretty hard to reach!

However, if you have a lot of implicit Tablespaces where no-one tidies up and DROPs the implicit database it can add up quickly!

**SpaceAssuranceExpert (SAX)** to the rescue!

SAX validates how near you are to the DB limit...

# Limits through design

Is there an Object limit?

Actually yes there is!

- In a Database you can have a maximum of 32,767 OBIDs (Object Ids). Now remember the OBID is not 1:1 for any and all objects.
- Each tablespace, index or referential relationship takes two, whereas each table, check constraint, aux for LOB, XML for XML, trigger or view with INSTEAD OF takes one.

So make sure you check these counts on a regular basis as well.

# Limits through design

Is there an Object limit?

Actually yes there is!

- In a Database you can have a maximum of 32,767 OBIDs (Object Ids). Now remember the OBID is not 1:1 for any and all objects.
- Each tablespace, index or referential relationship takes two, whereas each table, check constraint, aux for LOB, XML for XML, trigger or view with INSTEAD OF takes one.

So make sure you check these counts on a regular basis as well.

**SpaceAssuranceExpert** (SAX) to the rescue!

SAX validates how near you are to the cliff edge...

# Limits through design

Death by DBAT?

You all know what happens when you run out of **DBATs** right?

It is not pretty...

Just a **-DISPLAY DDF DETAIL** is all you need...

# Limits through design

Sadly not...

The problem here is that the DBAT counts that are output when you do a -DISPLAY are, of course, only the \*local\* counts...

With Datasharing you only see the data of the Member that you are directly connected to which is about as useful as a one legged man at an \*\*\*\* kicking contest...

# Limits through design

Here is an example output from a datasharing system:

```
DSNL080I  -SB11 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION                LUNAME                GENERICCLU
DSNL083I  xxxxxxxx                xxxxxxxx.xxxxxxxx  -NONE
DSNL084I  TCPSPORT=xxxx  SECPOR=0      RESPOR=xxxx  IPNAME=-NONE
DSNL085I  IPADDR=:xxx.xxx.x.xx
DSNL086I  SQL      DOMAIN=xxxx.fritz.box
DSNL086I  RESYNC  DOMAIN=xxxx.fritz.box
DSNL087I  ALIAS                PORT  SECPOR STATUS
DSNL088I  TEST110                0    0      STOPD
DSNL089I  MEMBER IPADDR=:xxx.xxx.x.xx
DSNL090I  DT=A  CONDBAT= 10000  MDBAT= 200
DSNL091I  MCONQN= 0  MCONQW= 0
DSNL092I  ADBAT= 0  QUEDBAT= 0  INADBAT= 0  CONQUED= 0
DSNL093I  DSCDBAT= 0  INACONN= 0
DSNL094I  WLMHEALTH=100  CLSDCONQN= 0  CLSDCONQW= 0
DSNL100I  LOCATION SERVER LIST:
DSNL101I  WT IPADDR                IPADDR
DSNL102I  32 :xxx.xxx.x.xx
DSNL102I  32 :xxx.xxx.x.xx
DSNL105I  CURRENT DDF OPTIONS ARE:
DSNL106I  PKGREL = COMMIT
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```



# Limits through design

So you have two choices...

- Write a little REXX that runs a round robin style of –DISPLAYs
- Get all warm and cuddly with IFI command processing to do it all in one call...

However you get the data, the interesting numbers are in the following two

lines of output: DSNL090I DT=A CONDBAT= 10000 **MDBAT**= 200  
DSNL092I **ADBAT**= 0 QUEDBAT= 0 INADBAT= 0 CONQUED= 0

The MDBAT is your **MAXDBAT** value and ADBAT is the current number **DBATs**.

You **\*must\*** monitor this all the time to see if any member is running out of **DBATs**!

# Limits through design

So you have **three** choices...

- Write a little REXX that runs a round robin style of –DISPLAYs
- Get all warm and cuddly with IFI command processing to do it all in one call...

However you get the data, the interesting numbers are in the following two lines of output:

```
DSNL090I DT=A CONDBAT= 10000 MDBAT= 200
DSNL092I ADBAT= 0 QUEDBAT= 0 INADBAT= 0 CONQUED= 0
```

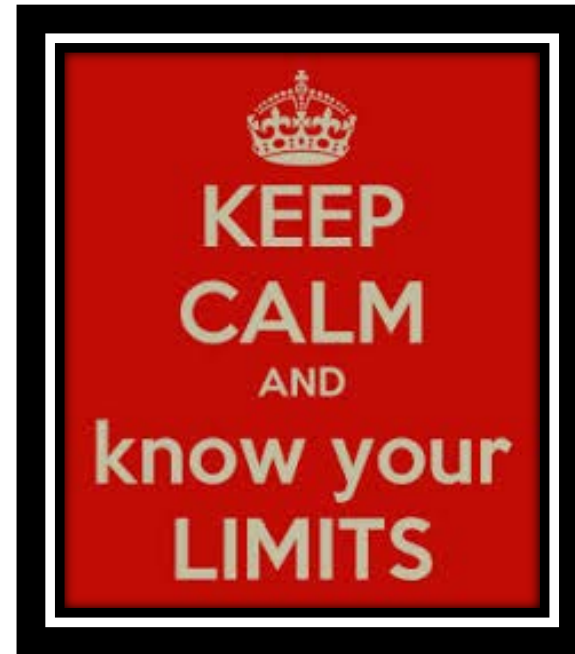
The MDBAT is your **MAXDBAT** value and ADBAT is the current number **DBATs**.

You **\*must\*** monitor this all the time to see if any member is running out of DBATs!

- **SpaceAssuranceExpert** (SAX) to the rescue!

# Agenda

- Limits through the machine
- Limits through logic
- Limits through design
- Future directions



# Future directions

IBM keep lifting the limits of Db2.

Now with **RPN** the last big bottleneck has been broken – You still must do a TS Level REORG with TP level inline image copies to migrate to it and you need a new Mapping table but when you are there it is a much better green than where you are standing now!

Will they ever raise the other limits?

I do not know of course, but I hope they raise the eligibility of **FITs** soon!

# Future directions

IBM keep lifting the limits of Db2.

Now with **RPN** the last big bottleneck has been broken – You still must do a TS Level REORG with TP level inline image copies to migrate to it and you need a new Mapping table but when you are there it is a much better green than where you are standing now!

Will they ever raise the other limits?

I do not know of course, but I hope they raise the eligibility of **FITs** soon!

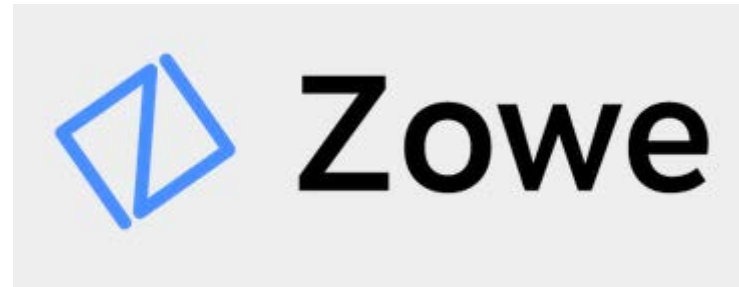
Whatever happens, simply remember:

**SpaceAssuranceExpert (SAX)** comes to the rescue!

# Our future directions

SEG keep lifting the limits of Db2.

We are now integrating all of our products into ZOWE and we are looking forward to 2020 with a relaunched range of ZOWE enabled products!





# Questions???

Many thanks for your attention and now....







Please fill out your session evaluation  
before leaving!

Session code: V09



**IDUG**

Leading the Db2 User  
Community since 1988