

IDUG

2026

Sydney | March 16 - 18

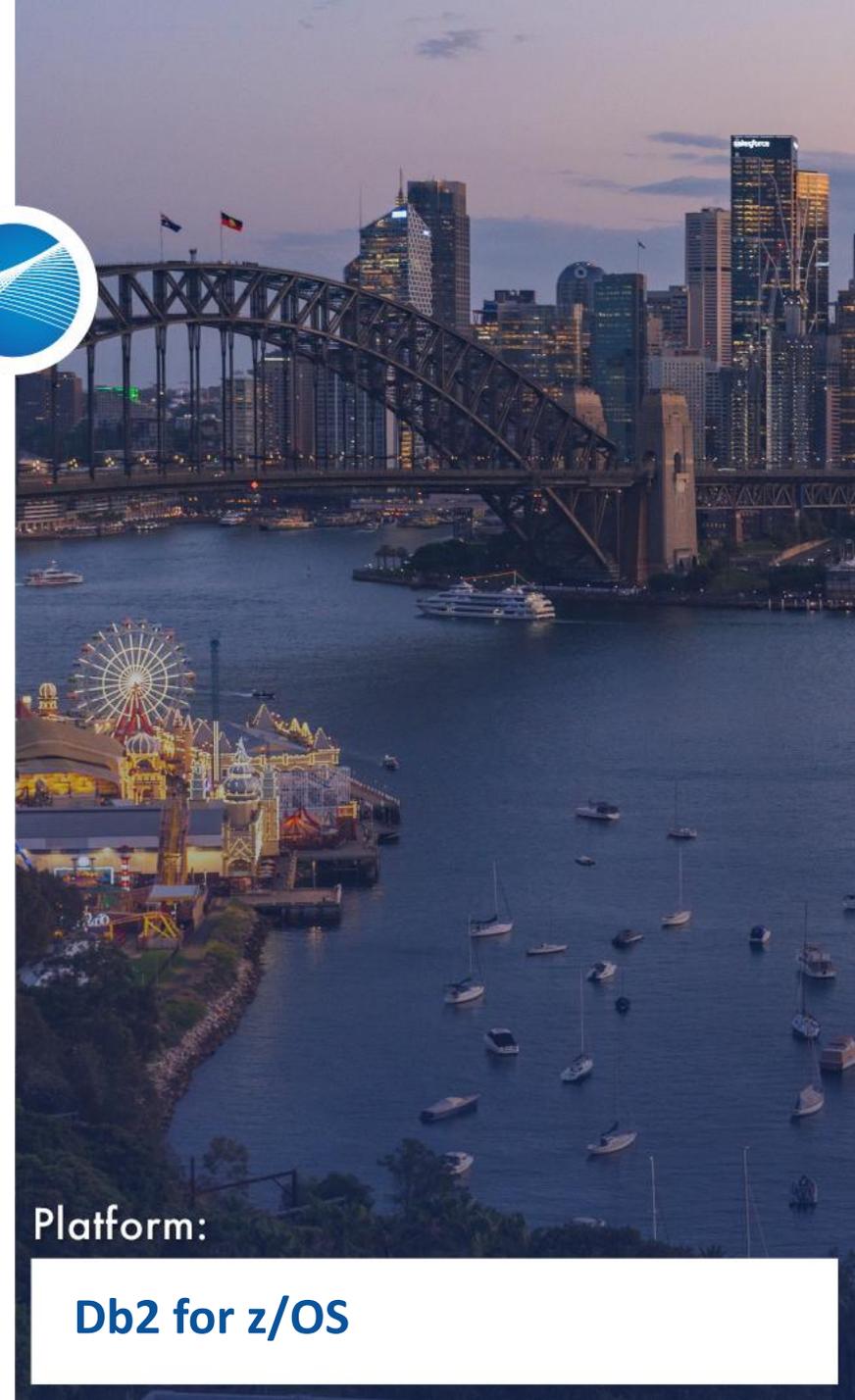
# AU Db2 TECH CONFERENCE

Autonomous Auditing of Db2  
Security Related Tables

Steen Rasmussen, *Broadcom*

---

Session Code: A11



Platform:

Db2 for z/OS

# ABSTRACT

---

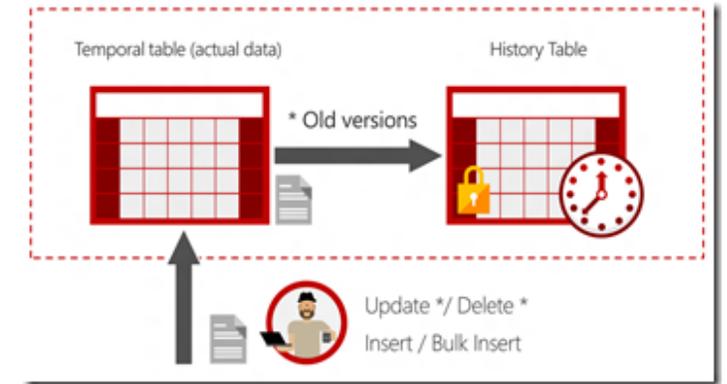
Are you challenged with tasks having to provide audit reports detailing who has access to which objects and who has which privileges – even worse, who had access 3 months ago ?

Many Db2 sites are using a log tool to accomplish these tasks – but are you aware of the relatively new capabilities in Db2 dealing with security information stored in the catalog ?

We will dive into what Db2 has to offer natively and how you can handle regulatory requirements in an easier way augmenting your current processes.

# AGENDA

- Db2 13 FL505 security related capabilities
- How do System Time Temporal tables work
- What is being recorded automatically
- Detailed walk through of some use cases using a live Db2 system
- Benefits, pros and cons – and some head scratching pieces



# Steen's IDUG Participation (including virtual ones)

---

- IDUG EMEA 1995-2025 (minus three) : 29
- IDUG NA 2000-2025 (minus 1) : 25
- APJ : 12 + 1 (this year)
- India : 3
- LATAM : 1
- Total 71



# Challenge to address for native Db2 security (1 | 3)

---

- The catalog only holds the **CURRENT** authorizations/permissions
- How do you satisfy auditing requirements ?
  - Not as big a challenge for external security
  - Save off / unload parts of the catalog ?
  - Using a Db2 LOG tool to generate daily reports ?
  - The LOG tools create great reports – but how easy is it to answer: “**Who had which permissions on table xxx.yyyy January 10 2025 ?**”

# Challenge to address for native Db2 security (2 | 3)

---

- Db2 13 FL505 introduced a new capability : Db2 Security using System Time Temporal tables (STT).
- The journey started with Db2 12
  - *STT columns added to BASE catalog tables – time travel not enabled*
  - Identical to RTS tables enabled in Db2 12
  - New tables created with same name but appended with **\_H** (e.g. *SYSDBAUTH* and *SYSDBAUTH\_H*)

# Challenge to address for native Db2 security (3 | 3)

- Db2 12 didn't allow enabling the STT versioning
- Db2 13 FL505+ is required – otherwise SQL-607

```
.CONNECT D13A
BPA0198I: CURRENT FUNCTION LEVEL IS V13R1M501
RETCODE = 0

.OPTION NOERRORS NOSQLERRORS NOLOG SQLFORMAT(SQL)
RETCODE = 0

ALTER TABLE SYSIBM.SYSDBAUTH
ADD VERSIONING USE HISTORY TABLE SYSIBM.SYSDBAUTH_H
ON DELETE ADD EXTRA ROW;
DSNT408I SQLCODE = -607, ERROR: OPERATION OR OPTION ADD VERSIONING IS
NOT DEFINED FOR THIS OBJECT
```

- SYSADM not sufficient to activate STT :
  - INSTALL SYSADM
  - SECADM another option

# Db2 Security Related Capability (1 | 5)

---

- Catalog tables enabled for System Time Temporal Table (versioning)

SYSAUDITPOLICIES  
SYSCOLAUTH  
SYSCONTEXTAUTHIDS  
SYSCONTEXT  
SYSCONTROLS  
SYSCONTROLS\_DESC  
SYSCONTROLS\_RTXT  
SYSCTXTTRUSTATTRS

SYSDBAUTH  
SYSPACKAUTH  
SYSPLANAUTH  
SYSRESAUTH  
SYSROLES  
SYSROUTINEAUTH  
SYSSCHEMAAUTH  
SYSSEQUENCEAUTH

SYSTABAUTH  
SYSUSERAUTH  
SYSVARIABLEAUTH  
SYSVIEWDEP

- Most make sense – but one odd ball .....

# Db2 Security Related Capability (2 | 5)

- Operations which will trigger STT actions

| Catalog table                     | History table             | Operations that can result in an update  |
|-----------------------------------|---------------------------|--|
| <a href="#">SYSAUDITPOLICIES</a>  | SYSIBM.SYSAUDITPOLICIES_H | UPDATE or DELETE an existing record  |
| <a href="#">SYSCOLAUTH</a>        | SYSIBM.SYSCOLAUTH_H       | TRANSFER OWNERSHIP<br>REVOKE table privileges<br>DROP TABLE or VIEW<br>DROP TRIGGER (instead of trigger)<br>RENAME TABLE |
| <a href="#">SYSCONTEXTAUTHIDS</a> | SYSIBM.SYSCONTEXTAUTHID_H | ALTER or DROP TRUSTED CONTEXT  |
| <a href="#">SYSCONTEXT</a>        | SYSIBM.SYSCONTEXT_H       | ALTER or DROP TRUSTED CONTEXT  |
| <a href="#">SYSCONTROLS</a>       | SYSIBM.SYSCONTROLS_H      | ALTER or DROP PERMISSION<br>ALTER or DROP MASK   |
| <a href="#">SYSCONTROLS_DESC</a>  | SYSIBM.SYSCONTROLS_DESC_H | ALTER or DROP PERMISSION<br>ALTER or DROP MASK   |
| <a href="#">SYSCONTROLS_RTXT</a>  | SYSIBM.SYSCONTROLS_RTXT_H | ALTER or DROP PERMISSION<br>ALTER or DROP MASK   |
| <a href="#">SYSCTXTRUSTATTRS</a>  | SYSIBM.SYSCTXTRUSTATTR_H  | ALTER or DROP TRUSTED CONTEXT  |

# Db2 Security Related Capability (3 | 5)

- Operations which will trigger STT actions

| Catalog table               | History table        | Operations that can result in an update   |
|-----------------------------|----------------------|---|
| <a href="#">SYSDBAUTH</a>   | SYSIBM.SYSDBAUTH_H   | TRANSFER OWNERSHIP<br>REVOKE database privileges<br>DROP DATABASE   |
| <a href="#">SYSPACKAUTH</a> | SYSIBM.SYSPACKAUTH_H | BIND or REBIND PACKAGE OWNER change<br>REVOKE package privileges<br>DROP or FREE PACKAGE<br>CREATE OR REPLACE PROCEDURE when replacing procedure<br>ALTER PROCEDURE (SQL native)<br>PACKAGE OWNER change<br>DROP PROCEDURE (SQL native)<br>ALTER FUNCTION (compiled SQL scalar)<br>PACKAGE OWNER change<br>DROP FUNCTION (compiled SQL scalar)<br>CREATE OR REPLACE TRIGGER (advanced) when replacing trigger<br>DROP TRIGGER |
| <a href="#">SYSPLANAUTH</a> | SYSIBM.SYSPLANAUTH_H | BIND or REBIND PLAN OWNER change<br>REVOKE plan privileges<br>FREE PLAN   |

# Db2 Security Related Capability (4 | 5)

- Operations which will trigger STT actions

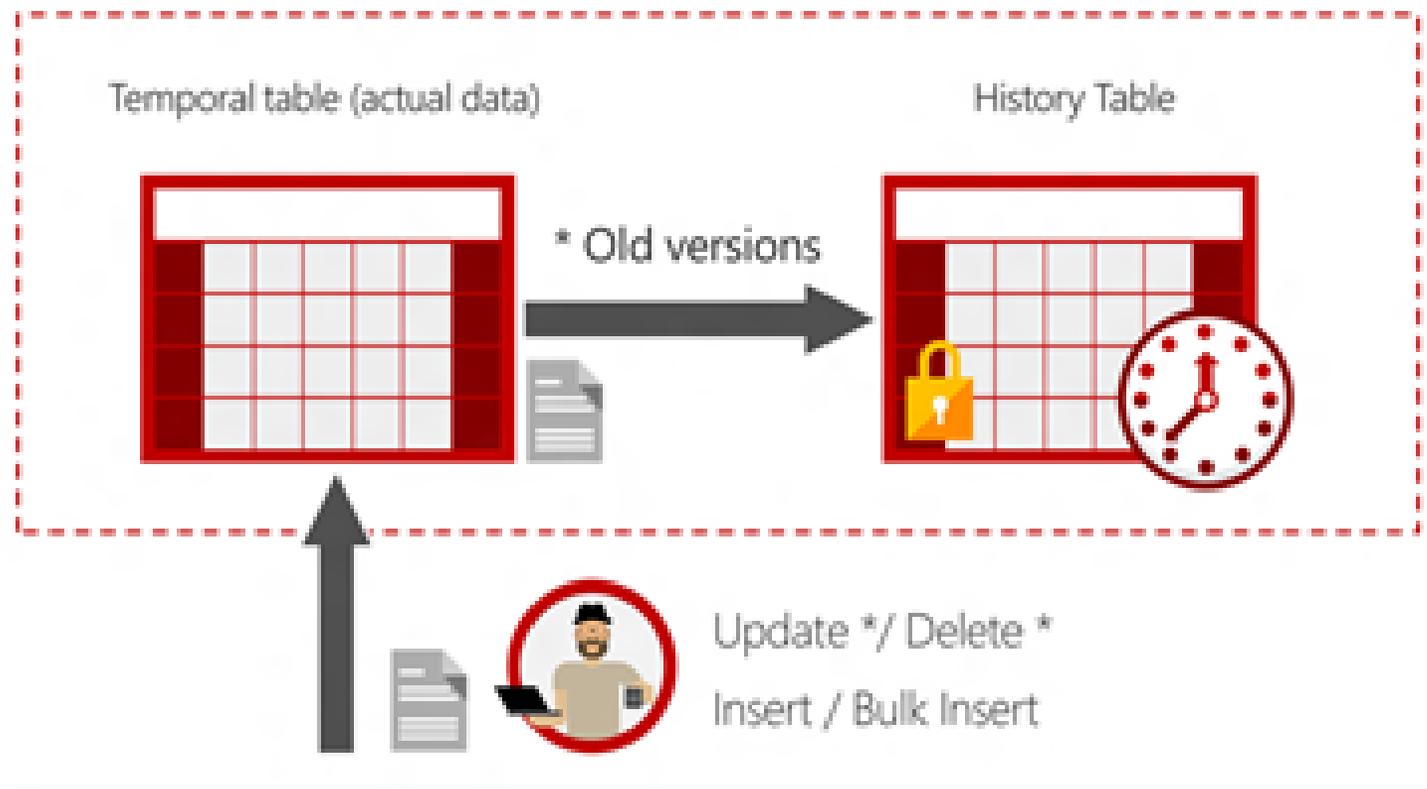
| Catalog table         | History table           | Operations that can result in an update  |
|-----------------------|-------------------------|--|
| <b>SYSRESAUTH</b>     | SYSIBM.SYSRESAUTH_H     | TRANSFER OWNERSHIP of STOGROUP<br>REVOKE <ul style="list-style-type: none"> <li>- Collection privileges</li> <li>- Type of JAR file privileges</li> <li>- USE of buffer pool</li> <li>- STOGROUP</li> <li>- Table space privileges</li> </ul> DROP Distinct Type, STOGROUP, or table space<br>SQLJ.REMOVE_JAR stored procedure |
| <b>SYSROLES</b>       | SYSIBM.SYSROLES_H       | DROP ROLE  |
| <b>SYSROUTINEAUTH</b> | SYSIBM.SYSROUTINEAUTH_H | REVOKE function or procedure privileges<br>CREATE or REPLACE PROCEDURE<br>DROP FUNCTION or PROCEDURE   |
| <b>SYSSCHEMAAUTH</b>  | SYSIBM.SYSSCHEMAAUTH_H  | REVOKE schema privileges   |

# Db2 Security Related Capability (5 | 5)

- Operations which will trigger STT actions

| Catalog table                   | History table            | Operations that can result in an update  |
|---------------------------------|--------------------------|--|
| <a href="#">SYSSEQUENCEAUTH</a> | SYSIBM.SYSSEQUENCEAUTH_H | REVOKE sequence privileges<br>DROP SEQUENCE  |
| <a href="#">SYSTABAUTH</a>      | SYSIBM.SYSTABAUTH_H      | TRANSFER OWNERSHIP<br>REVOKE table privileges<br>DROP TABLE or VIEW<br>DROP TRIGGER (instead of trigger)<br>RENAME TABLE |
| <a href="#">SYSUSERAUTH</a>     | SYSIBM.SYSUSERAUTH_H     | REVOKE system privileges   |
| <a href="#">SYSVARIABLEAUTH</a> | SYSIBM.SYSVARIABLEAUTH_H | REVOKE variable privileges<br>DROP VARIABLE  |

# What is a System Time Temporal Table (1 | 4)



# What is a System Time Temporal Table (2 | 4)

---

- Also known as Time Travel – what did the row look like at a specific point in time
- BASE TABLE (the “real” table with current data)
- HISTORY table associated to BASE
- Specific column attributes
- INSERT into BASE nothing happens to HISTORY
- BASE updated/deleted -> old “look and feel” inserted into history including timestamp from BASE insert and BASE update.
- Special SQL extensions to retrieve data – or you can use normal SQL (*more later*)

# What is a System Time Temporal Table (3 | 4)

- Characteristics for Base tables needed to enable STT
  - Specific columns/attributes to be included in both BASE & HISTORY table

```
,SYS_START    TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS ROW BEGIN
,SYS_END      TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS ROW END
,TRANS_START  TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS TRANSACTION START ID
,PERIOD       FOR SYSTEM_TIME (SYS_START , SYS_END)
```

- The same columns needed for the associated History table
- PERIOD attribute described the timestamps for when the row was "active"
- Querying the history table will illustrate the "look and feel" over time
- ALTER statement executed to ENABLE VERSIONING

```
ALTER TABLE base-table ADD VERSIONING USE HISTORY TABLE history-table ON DELETE ADD EXTRA ROW
```

# What is a System Time Temporal Table (4 | 4)

- Why might you want to enable **ON DELETE ADD EXTRA ROW** – CHATGPT provided a great answer

## Purpose of ON DELETE ADD EXTRA ROW

When a row is **deleted** from a **system-versioned temporal table**, Db2 must archive the deleted row into the **history table** to maintain the full timeline of changes.

By default, Db2 will simply record the deleted row with an adjusted `SYSTEM_END` timestamp to mark when the row became invalid.

But when you specify:

```
sql
```

 Copy code

```
ON DELETE ADD EXTRA ROW
```

Db2 will **add an additional row** to the **history table** to explicitly record the **deletion event** — making it easier to audit deletions separately.

## When to Use It

Use `ON DELETE ADD EXTRA ROW` when you want to **distinguish a deletion** from an update.

Without it, a delete might look like just another update (where the `SYSTEM_END` time is changed). With it, you

Ask anything

 Attach

 Search

 Study

 Voice

# Use Cases – How it Works

## Use Case 1 (1|2)

- GRANT DROP ON DATABASE IDUGDB01 TO USER2
  - SYSDBAUTH\_H is empty
- REVOKE DROP ON DATABASE IDUGDB01 FROM USER2
  - Two rows inserted into HISTORY (*due to **ON DELETE ADD EXTRA ROW** : START and END time identical for delete while DROP illustrates when the row "lived"*)

```

2 ROWS RETRIEVED
SYS_START                               SYS_END
2025-08-07-17.33.34.994639102000 2025-08-07-17.35.30.143030100000
2025-08-07-17.35.30.143030100000 2025-08-07-17.35.30.143030100000
***** BOTTOM OF DATA *****
    
```

# Use Case 1 (2|2)

- The DROP initially granted is no longer is SYSDBAUTH but now sits in SYSDBAUTH\_H

```

For Table => SYSIBM.SYSDBAUTH_H > Row number=> 1 OF 2
Browse Mode => F Max Char => 256
SSID: AD22 -----FETCH STATUS: COMPLETE-----
##.COLUMN NAME NULL DATA FOR ROW # 1
A1.GRANTOR RASST02
A2.GRANTEE USER2
A3.NAME IDUGDB01
A4.TIMESTAMP ES5BVYLP HH6P
A5.DATEGRANTED 250807
A6.TIMEGRANTED 17333499
A7.GRANTEETYPE
A8.DROPAUTH Y
A9.GRANTEDTS 2025-08-07-17.33.34.994563
B1.GRANTORTYPE
B2.SYS_START 2025-08-07-17.33.34.994639102000
B3.SYS_END 2025-08-07-17.35.30.143030100000
B4.TRANS_START 2025-08-07-17.33.34.994639102000
B5.GEN_SESSION_USER N RASST02
***** BOTTOM OF DATA *****
  
```

```

For Table => SYSIBM.SYSDBAUTH_H > Row number=> 2 OF 2
Browse Mode => F Max Char => 256
SSID: AD22 -----FETCH STATUS: COMPLETE-----
##.COLUMN NAME NULL DATA FOR ROW # 2
A1.GRANTOR RASST02
A2.GRANTEE USER2
A3.NAME IDUGDB01
A4.TIMESTAMP ES5BVYLP HH6P
A5.DATEGRANTED 250807
A6.TIMEGRANTED 17333499
A7.GRANTEETYPE
A8.DROPAUTH Y
A9.GRANTEDTS 2025-08-07-17.33.34.994563
B1.GRANTORTYPE
B2.SYS_START 2025-08-07-17.35.30.143030100000
B3.SYS_END 2025-08-07-17.35.30.143030100000
B4.TRANS_START 2025-08-07-17.33.34.994639102000
B5.GEN_SESSION_USER N RASST02
***** BOTTOM OF DATA *****
  
```

## Use Case 2 (1|4)

- Change OWNERSHIP of database – initially RASST02
- First to **IDUG** – then to **IDUGNA**
- Second TRANSFER requires SET CURRENT SQLID = **IDUG**

```
.CONNECT AD22
BPA0198I: CURRENT FUNCTION LEVEL IS V13R1M507
RETCODE = 0

.OPTION NOERRORS NOSQLERRORS NOLOG SQLFORMAT(SQL)
RETCODE = 0

TRANSFER OWNERSHIP OF DATABASE IDUGDB01
TO USER IDUG REVOKE PRIVILEGES ;
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
```

```
.CONNECT AD22
BPA0198I: CURRENT FUNCTION LEVEL IS V13R1M507
RETCODE = 0

.OPTION NOERRORS NOSQLERRORS NOLOG SQLFORMAT(SQL)
RETCODE = 0

TRANSFER OWNERSHIP OF DATABASE IDUGDB01 TO USER IDUGNA REVOKE PRIVILEGES

;
DSNT408I SQLCODE = -551, ERROR: RASST02 DOES NOT HAVE THE PRIVILEGE TO
PERFORM OPERATION TRANSFER OWNER ON OBJECT IDUGDB01
DSNT418I SQLSTATE = 42501 SQLSTATE RETURN CODE
DSNT415I SQLERRP = DSNXATRO SQL PROCEDURE DETECTING ERROR
```

# Use Case 2 (2|4)

```

RUBROWF ----- RC/Browse: Form Mode -----
COMMAND ==> █

For Table => SYSIBM.SYSDBAUTH_H > Row nu
Browse Mode => F Max Ch

SSID: AD22 -----FETCH STATUS: COMPLETE-----
##.COLUMN NAME NULL DATA FOR ROW # 5
A1.GRANTOR RASST02
A2.GRANTEE RASST02
A3.NAME IDUGDB01
A4.TIMESTAMP ES416KDAYJSK
A5.DATEGRANTED 250806
A6.TIMEGRANTED 17474210
A7.GRANTEETYPE
A8.AUTHHOWGOT
A9.CREATETABAUTH G
B1.CREATETSAUTH G
B2.DBADMAUTH G
B3.DBCTRLAUTH G
B4.DBMAINTAUTH G
B5.DISPLAYDBAUTH G
B6.DROPAUTH G
B7.IMAGCOPYAUTH G
B8.LOADAUTH G
B9.REORGAUTH G
C1.RECOVERDBAUTH G
C2.REPAIRAUTH G
C3.STARTDBAUTH G
C4.STATSAUTH G
C5.STOPAUTH G
C6.IBMREQD N
C7.GRANTEDTS 2025-08-06-17.47.42.109369
    
```

Re

```

RUBROWF ----- RC/Browse: Form Mode -----
COMMAND ==> █

For Table => SYSIBM.SYSDBAUTH_H > Row nu
Browse Mode => F Max Ch

SSID: AD22 -----FETCH STATUS: COMPLETE-----
##.COLUMN NAME NULL DATA FOR ROW # 6
A1.GRANTOR IDUG
A2.GRANTEE IDUG
A3.NAME IDUGDB01
A4.TIMESTAMP ETERLP7GATP9
A5.DATEGRANTED 250910
A6.TIMEGRANTED 09372587
A7.GRANTEETYPE
A8.AUTHHOWGOT
A9.CREATETABAUTH G
B1.CREATETSAUTH G
C9.SYS_START 2025-08-07-16.48.53.558787734000
D1.SYS_END 9999-12-30-00.00.00.000000000000
D2.TRANS_START 2025-08-07-16.48.53.558787734000
D3.GEN SESSION USER N RASST02
B8.LOADAUTH G
B9.REORGAUTH G
C1.RECOVERDBAUTH G
C2.REPAIRAUTH G
C3.STARTDBAUTH G
C4.STATSAUTH G
C5.STOPAUTH G
C6.IBMREQD N
C7.GRANTEDTS 2025-09-10-09.37.25.877787
    
```



## Use Case 2 (3|4)

```
SELECT * FROM SYSIBM.SYSDBAUTH WHERE NAME = 'IDUGDB01'
FOR SYSTEM TIME BETWEEN
'2025-08-01-00.00.00.000000000000' AND
'2025-10-01-00.00.00.000000000000'
```

```
DSNT408I  SQLCODE = -199, ERROR:  ILLEGAL USE OF KEYWORD SYSTEM TIME.
        TOKEN FETCH UPDATE READ SINGLE MULTIPLE WAS EXPECTED
```

```
SELECT * FROM SYSIBM.SYSDBAUTH
FOR SYSTEM TIME BETWEEN
'2025-08-01-00.00.00.000000000000' AND
'2025-10-01-00.00.00.000000000000'
WHERE NAME = 'IDUGDB01' ;
DSNT400I  SQLCODE = 000, SUCCESSFUL EXECUTION
```

One “odd” discovery:  
How SQL is coded DOES  
matter ...

Transfer OWNERSHIP  
execution – what happens ?

## Use Case 2 (4|4)

```
UPDATE "SYSIBM".SYSDATABASE
SET "NAME" = 'IDUGDB01'
, CREATOR = 'IDUG'
WHERE "NAME" = 'IDUGDB01';

INSERT INTO "SYSIBM".SYSDBAUTH_H
( GRANTOR , GRANTEE , "NAME" , TIMESTAMP , DATEGRANTED ,
. . . . .,
SYS_START , SYS_END , TRANS_START , GEN_SESSION_USER )
VALUES
( 'RASST02' , 'RASST02' , 'IDUGDB01' ,
X'C5E2F4F1F6D2C4C1E8D1E2D2' , '250806' , '17474210' , '' ,
' ' , 'G' ,
'G' , 'G' , 'G' , 'G' , 'G' , 'G' , 'N' ,
'2025-08-06-17.47.42.109369' , ' ' ,
'2025-08-06-17.47.42.109396667000' ,
'2025-09-10-09.37.25.877821626000' ,
'2025-08-06-17.47.42.109396667000' , 'RASST02' )
;
```

```
UPDATE "SYSIBM".SYSDBAUTH
SET GRANTOR = 'IDUG'
, GRANTEE = 'IDUG'
, TIMESTAMP =
, DATEGRANTED = '250910'
, TIMEGRANTED = '09372587'
, GRANTEDTS = '2025-09-10-09.37.25.877787'
WHERE GRANTOR = 'RASST02'
AND GRANTEE = 'RASST02'
AND "NAME" = 'IDUGDB01'
. . . . .
AND GEN_SESSION_USER = 'RASST02'
```

```
-----
-- SQL Generation Summary Statistics:
-- -----
-- Number of INSERT statements generated:      1
-- Number of DELETE statements generated:      0
-- Number of UPDATE statements generated:      2
```

# Use Case 3 (1|4)

- Tracking Table Authorizations and REVOKEs

```
RQTUA          ----- RC/Q Table User Authorization Inq ----- 2025/10/09 12:38
COMMAND ==> █                                     SCROLL ==> CSR

DB2 Object ==> T                                Option ==> UA   Where => N
Table Name ==> XFEROWN1                          > Creator ==> RASST02
Qualifier ==> *                                  > Grantor ==> *
Loc: LOCAL ----- SSID: AD22  LVL: 02 -RASST02 -          LINE 1 OF 7  >
CMD          TABLE/USER      SEL CTR/GRNTOR  DATE      TIME      ALT DEL INDX
-----
XFEROWN1
RASST02      G    RASST02   2025/09/10 02:25:59  G    G    G
USER01       Y    RASST02   2025/09/10 06:32:56
USER02       Y    RASST02   2025/09/10 06:32:56
USER03       G    RASST02   2025/09/10 06:32:56
USER04       Y    USER03    2025/09/10 06:32:56
USER05       Y    USER03    2025/09/10 06:32:56
***** BOTTOM OF DATA *****
```

```
CREATE TABLE RASST02.XFEROWN1
(COL01 CHAR (5) NOT NULL WITH DEFAULT,
 COL02 SMALLINT NOT NULL WITH DEFAULT)
IN IDUGDB01.IDUGTS02;
```

```
GRANT SELECT ON TABLE RASST02.XFEROWN1 TO USER01;
GRANT SELECT ON TABLE RASST02.XFEROWN1 TO USER02;
GRANT SELECT ON TABLE RASST02.XFEROWN1 TO USER03
with grant option;
```

```
SET CURRENT SQLID = 'USER03' ;
GRANT SELECT ON TABLE RASST02.XFEROWN1 TO USER04;
GRANT SELECT ON TABLE RASST02.XFEROWN1 TO USER05;
```

## Use Case 3 (2|4)

---

- Tracking Table Authorizations and REVOKEs

```
REVOKE SELECT ON TABLE RASST02.XFEROWN1 FROM USER02 BY RASST02;  
REVOKE SELECT ON TABLE RASST02.XFEROWN1 FROM USER04 BY USER03;
```

## Use Case 3 (3 | 4)

- Interestingly observation:
- The TWO REVOKEs are inserted into the history table (*REVOKE executed October 09*)
- BUT – also what lead to how these ID's got the permissions initially.
- This is where SQL extension gets handy



| GRANTOR | GRANTEE | SYS_START                        | SYS_END                          |
|---------|---------|----------------------------------|----------------------------------|
| RASST02 | RASST02 | 2025-09-10-02.25.59.090489904000 | 9999-12-30-00.00.00.000000000000 |
| RASST02 | USER01  | 2025-09-10-06.32.56.094359165000 | 9999-12-30-00.00.00.000000000000 |
| RASST02 | USER03  | 2025-09-10-06.32.56.094359165000 | 9999-12-30-00.00.00.000000000000 |
| USER03  | USER05  | 2025-09-10-06.32.56.094359165000 | 9999-12-30-00.00.00.000000000000 |
| RASST02 | USER02  | 2025-09-10-06.32.56.094359165000 | 2025-10-09-12.41.01.359714833000 |
| USER03  | USER04  | 2025-09-10-06.32.56.094359165000 | 2025-10-09-12.41.45.489887957000 |

- Table AUTH retrieved using AS OF October 09

## Use Case 3 (4|4)

- When using AS OF October 10 instead of October 09 :
  - USER02 and USER04 are not included in the result set

```
SELECT * FROM SYSIBM.SYSTABAUTH FOR SYSTEM_TIME AS OF
'2025-10-10-00.00.00.000000'
WHERE TTNAME LIKE 'XFER%'
```

4 ROWS RETRIEVED

| GRANTOR | GRANTEE | GRANTEETYPE | DBNAME |
|---------|---------|-------------|--------|
| RASST02 | RASST02 |             |        |
| RASST02 | USER01  |             |        |
| RASST02 | USER03  |             |        |
| USER03  | USER05  |             |        |

\*\*\*\*\* BOTTOM OF DATA \*\*\*\*\*

# Do you still need a LOG tool for audit ?

---

- Maybe not ?
- All CURRENT permissions are in the normal catalog tables
- All DCL operations are recorded when CHANGED in the history tables
- But .....
- Generating WHAT was GRANT'ed / REVOKE'd (the exact statement) is a little harder
- What are the requirements to report ?
- Maybe a mix of both is beneficial.
- Documenting permissions for a specific user – or who had what access AS OF a specific date (even date range)
  - STT for security related tables can definitely help

# SQL extensions – how to access

---

- You can still use “normal” SQL to select from both base and history tables.
- Use the power of Temporal SQL extensions
  - FOR SYSTEM\_TIME using FROM BASE table
    - AS OF a specific point-in-time
    - BETWEEN for a time range
  - Db2 will UNION the BASE and HISTORY
- You can access each table WITHOUT extension as if they are “normal tables”.
- You can't UPDATE/DELETE history -> ALTER BASE REMOVE VERSIONING

# Concerns

---

- All tablespaces have MAXPARTITIONS 1
  - You can't ALTER catalog objects increasing MAXPARTITIONS
  - If a lot of activity consider cleaning up HISTORY table via DELETE or REORG DISCARD (*where SYS\_END < CURRENT\_TIMESTAMP - 12 MONTHS*)
- ON DELETE ADD EXTRA ROW isn't mandatory but you probably need this to audit when "permission" was removed/revoked.
- LISTDEF – remember to include HISTORY
- Recovery considerations – have to recover both
- Need to update HISTORY – ALTER REMOVE VERSIONING

IDUG

2026

Sydney | March 16 - 18

# AU Db2 TECH CONFERENCE

Autonomous Auditing of Db2

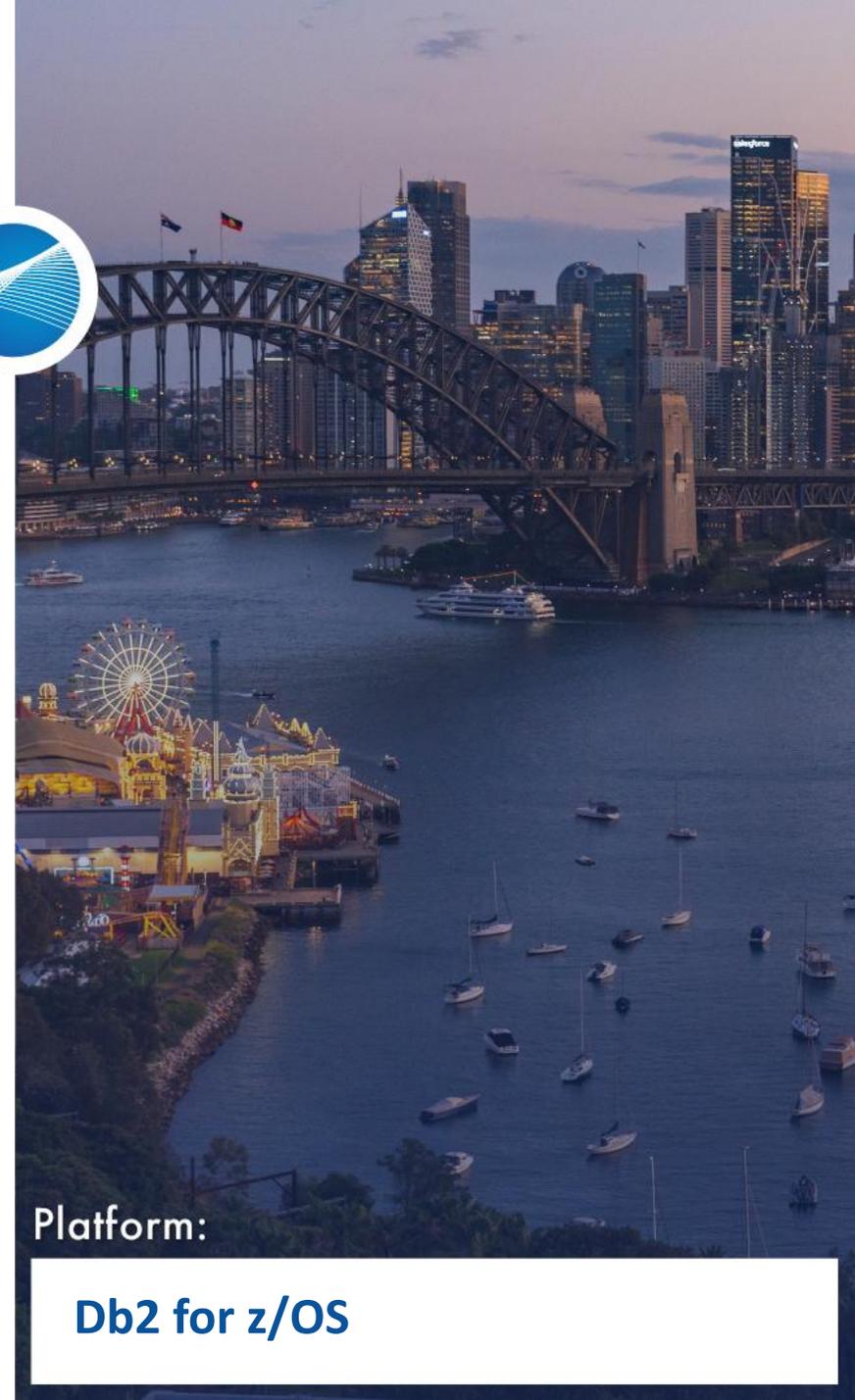
Security Related Tables

Steen Rasmussen, *Broadcom*

Contact: [steen.rasmussen@broadcom.com](mailto:steen.rasmussen@broadcom.com) / [db2steen@yahoo.com](mailto:db2steen@yahoo.com)

---

Session Code: A11



Platform:

Db2 for z/OS

An aerial view of Sydney, Australia, at dusk. The Sydney Harbour Bridge is prominent on the left, spanning the water. The city skyline is visible in the background with many skyscrapers lit up. In the foreground, there are many sailboats in the water and a theme park area with a Ferris wheel and roller coaster on the left. The sky is a mix of blue and orange from the setting sun.

# IDUG

2026 Australia **Db2** Tech Conference