Autonomous Auditing of Db2 Security Related Tables

Steen Rasmussen, Client Services Consultant, Broadcom

steen.rasmussen@broadcom.com / db2steen@yahoo.com

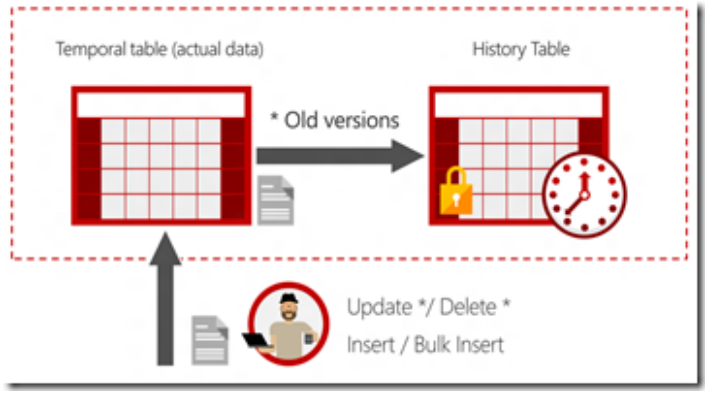
ABSTRACT

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 : 13
- 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 tbcr.tbnm January 10 2026 ?”***

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

- Db2 13 FL505 introduced a new capability : Db2 Security features 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
SYSAUDITPOLICIES	SYSIBM.SYSAUDITPOLICIES_H	UPDATE or DELETE an existing record
SYSCOLAUTH	SYSIBM.SYSCOLAUTH_H	TRANSFER OWNERSHIP REVOKE table privileges DROP TABLE or VIEW DROP TRIGGER (instead of trigger) RENAME TABLE
SYSCONTEXTAUTHIDS	SYSIBM.SYSCONTEXTAUTHID_H	ALTER or DROP TRUSTED CONTEXT
SYSCONTEXT	SYSIBM.SYSCONTEXT_H	ALTER or DROP TRUSTED CONTEXT
SYSCONTROLS	SYSIBM.SYSCONTROLS_H	ALTER or DROP PERMISSION ALTER or DROP MASK
SYSCONTROLS_DESC	SYSIBM.SYSCONTROLS_DESC_H	ALTER or DROP PERMISSION ALTER or DROP MASK
SYSCONTROLS_RTXT	SYSIBM.SYSCONTROLS_RTXT_H	ALTER or DROP PERMISSION ALTER or DROP MASK
SYSCTXTTRUSTATTRS	SYSIBM.SYSCTXTTRUSTATTR_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
SYSDBAUTH	SYSIBM.SYSDBAUTH_H	TRANSFER OWNERSHIP REVOKE database privileges DROP DATABASE
SYSPACKAUTH	SYSIBM.SYSPACKAUTH_H	BIND or REBIND PACKAGE OWNER change REVOKE package privileges DROP or FREE PACKAGE CREATE OR REPLACE PROCEDURE when replacing procedure ALTER PROCEDURE (SQL native) PACKAGE OWNER change DROP PROCEDURE (SQL native) ALTER FUNCTION (compiled SQL scalar) PACKAGE OWNER change DROP FUNCTION (complied SQL scalar) CREATE OR REPLACE TRIGGER (advanced) when replacing trigger DROP TRIGGER
SYSPLANAUTH	SYSIBM.SYSPLANAUTH_H	BIND or REBIND PLAN OWNER change REVOKE plan privileges FREE PLAN

Db2 Security Related Capability (4|5)

- Operations which will trigger STT actions

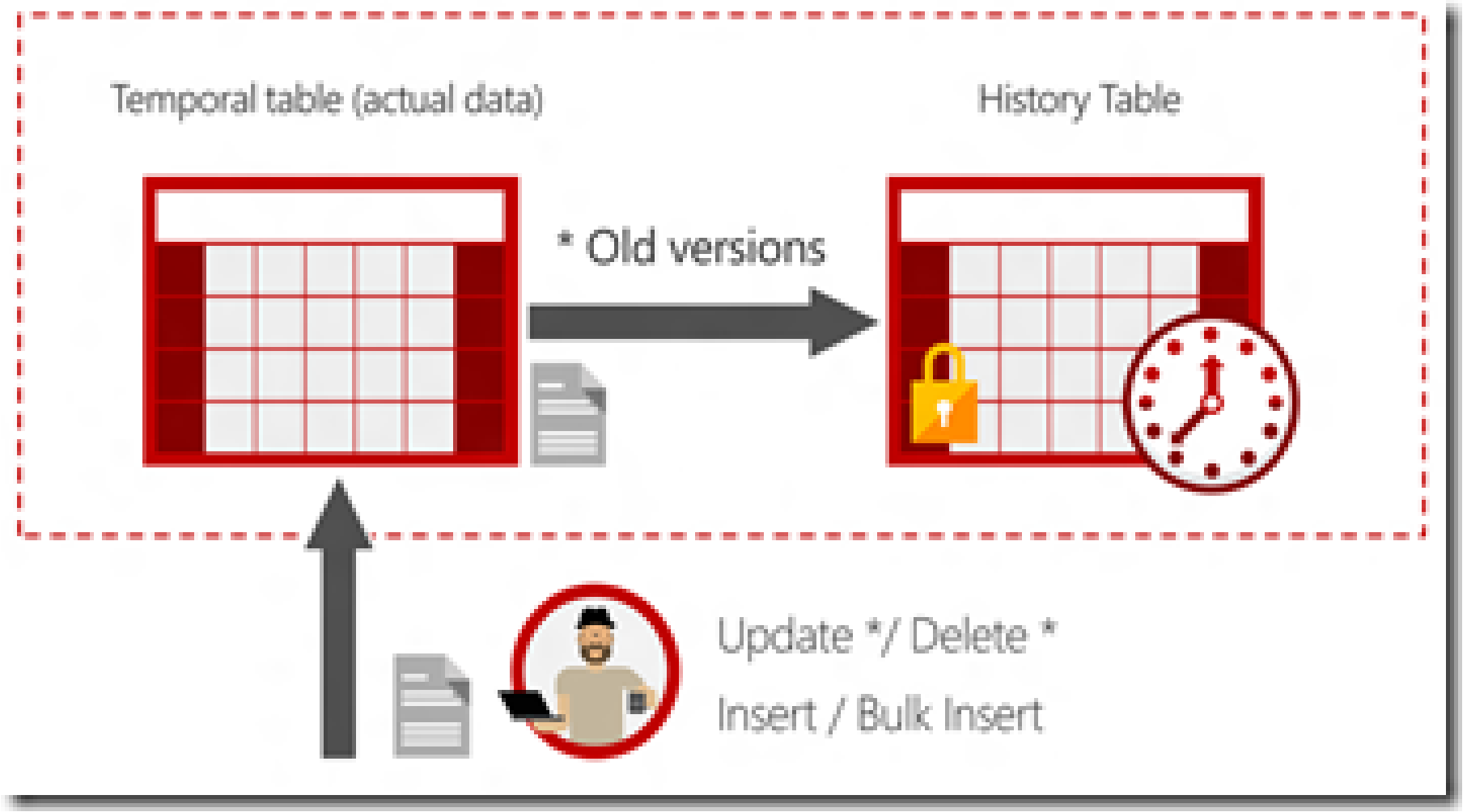
Catalog table	History table	Operations that can result in an update
SYSRESAUTH	SYSIBM.SYSRESAUTH_H	TRANSFER OWNERSHIP of STOGROUP REVOKE - Collection privileges - Type of JAR file privileges - USE of buffer pool - STOGROUP - Table space privileges DROP Distinct Type, STOGROUP, or table space SQLJ.REMOVE_JAR stored procedure
SYSROLES	SYSIBM.SYSROLES_H	DROP ROLE
SYSROUTINEAUTH	SYSIBM.SYSROUTINEAUTH_H	REVOKE function or procedure privileges CREATE or REPLACE PROCEDURE DROP FUNCTION or PROCEDURE
SYSSCHEMAAUTH	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
SYSSEQUENCEAUTH	SYSIBM.SYSSEQUENCEAUTH_H	REVOKE sequence privileges DROP SEQUENCE
SYSTABAUTH	SYSIBM.SYSTABAUTH_H	TRANSFER OWNERSHIP REVOKE table privileges DROP TABLE or VIEW DROP TRIGGER (instead of trigger) RENAME TABLE
SYSUSERAUTH	SYSIBM.SYSUSERAUTH_H	REVOKE system privileges
SYSVARIABLEAUTH	SYSIBM.SYSVARIABLEAUTH_H	REVOKE variable privileges 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 (when the row was “alive”).
- 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

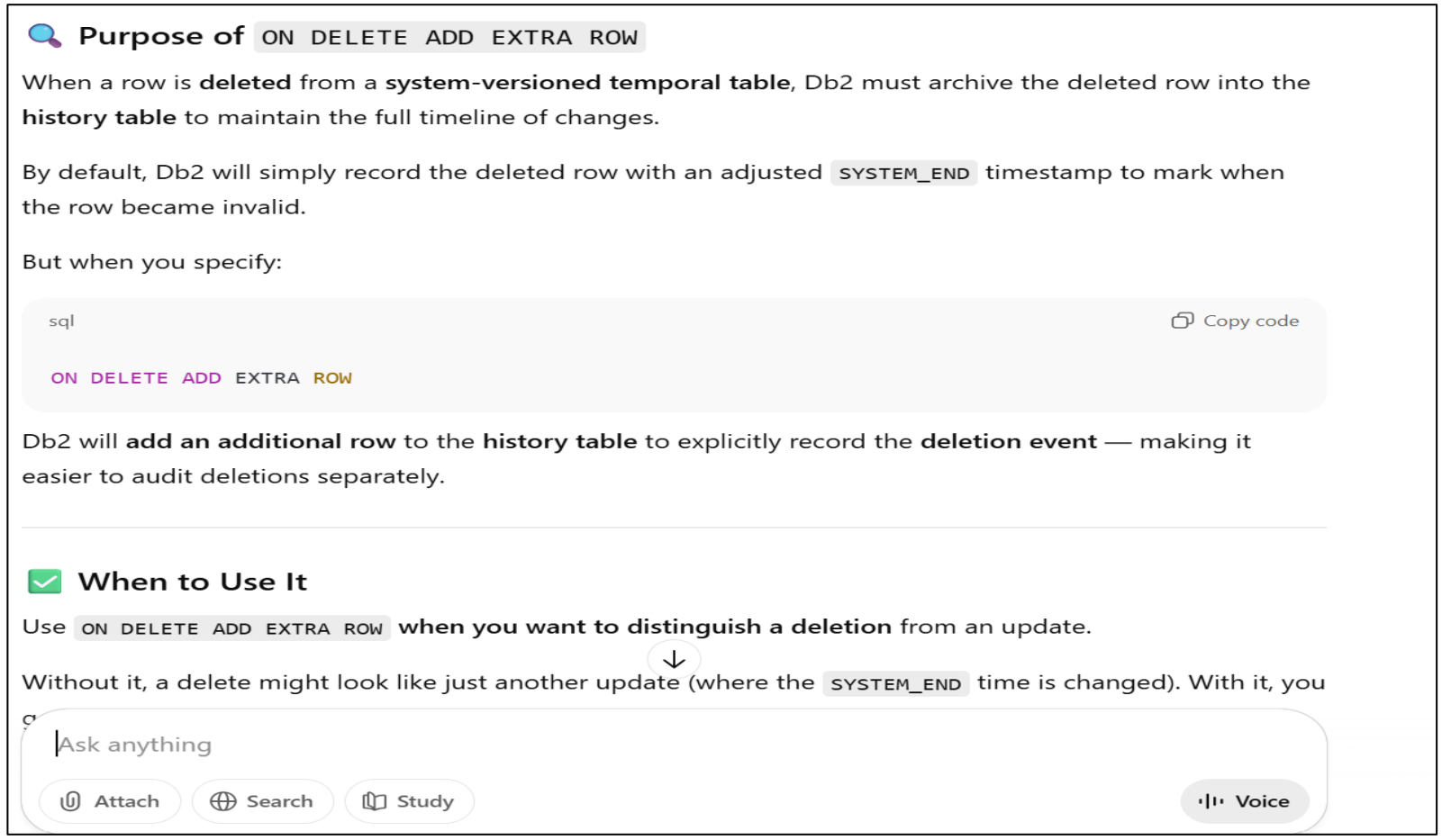
```
,SYS_START   TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS ROW BEGIN
,SYS_END     TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS ROW END
,TRANS_START TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL GENERATED ALWAYS AS TRANSACTION START ID
,PERIOD      FOR SYSTEM_TIME  (SYS_START , 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                ES5BVYLPHH6P
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                ES5BVYLPHH6P
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)

- Browse SYSDBAUTH_H : Reflects the two TRANSFER OWNERSHIPs

```
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
```

```
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
B2.DBADMAUTH                   G
B3.DBCTRLAUTH                  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
```

```
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
```

Use Case 2 (3|4)

- SQL extension(s) : how to retrieve data

```
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 …

Use Case 2 (4|4)

Transfer OWNERSHIP execution – what happens ?

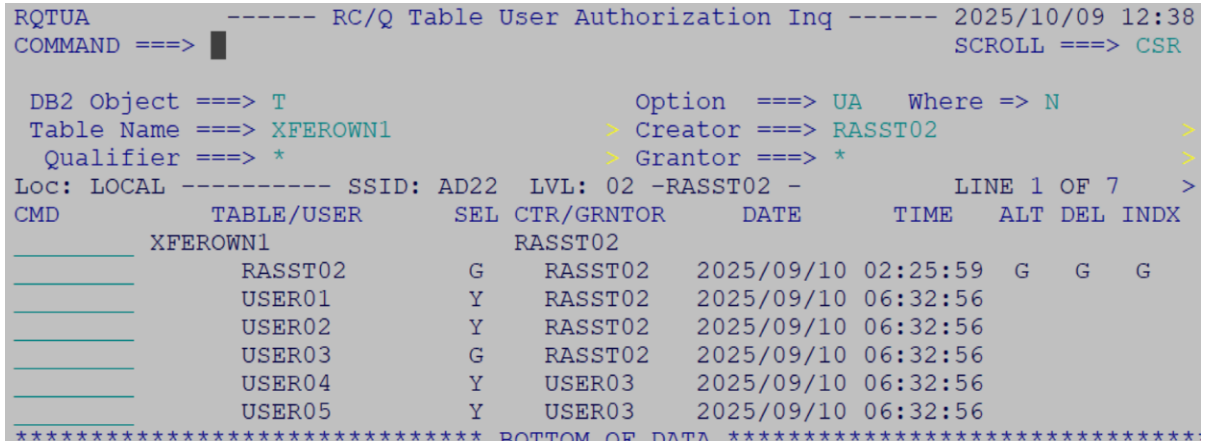
```
UPDATE "SYSIBM".SYSDATABASE
 SET "NAME" =   'IDUGDB01'
  , CREATOR =  'IDUG'
 WHERE "NAME" =   'IDUGDB01';

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' , '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




```
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

```
SYS_START                          SYS_END

2025-09-10-02.25.59.090489904000 9999-12-30-00.00.00.000000000000
2025-09-10-06.32.56.094359165000 9999-12-30-00.00.00.000000000000
2025-09-10-06.32.56.094359165000 9999-12-30-00.00.00.000000000000
2025-09-10-06.32.56.094359165000 9999-12-30-00.00.00.000000000000
2025-09-10-06.32.56.094359165000 2025-10-09-12.41.01.359714833000
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

Security Cleanup Concerns Remediation (1|3)

- Scared about REVOKE CASCADE challenges

- Relatively new native Db2 features can come to the rescue:
  - TRANSFER OWNERSHIP
  - REVOKE_DEP_PRIVILEGES
- Really concerned: Use a LOG TOOL to REDO REVOKE/TRANSFER after executing command using AUTOCOMMIT NO.

Security Cleanup Concerns Remediation (2|3)

- TRANSFER OWNERSHIP in action:
  - STEEN01 created DB=KDB01

```
TRANSFER OWNERSHIP OF DATABASE KDB01
TO USER SACHIN revoke privileges ;
```

```
 DB2 Object ===> DB                      Option  ===> L
  Data Base ===> KDB%                  > Creator ===> *
  Qualifier ===> *                     > N/A     ===> *
Loc: LOCAL ---------- SSID:      ---------- STEEN01 -
CMD      DATABASE CREATOR  STOGROUP CREATEDB   DBID BPOOL
________ KDB01    SACHIN   SYSDEFLT STEEN01   15042 BP0
****************************** BOTTOM OF DATA ************
```

```
UPDATE "SYSIBM".SYSDATABASE
   SET CREATOR = 'SACHIN'
   WHERE "NAME" = 'KDB01'    ;

UPDATE "SYSIBM".SYSDBAUTH
   SET GRANTOR = 'SACHIN'
     , GRANTEE = 'SACHIN'
     , TIMESTAMP = X'C5D4D8D1E4C7F9E4F2E3C9C8'
     , DATEGRANTED = '230602'
     , TIMEGRANTED = '08252643'
     , GRANTEDTS = '2023-06-02-08.25.26.435675'
   WHERE GRANTOR = 'STEEN01'
     AND GRANTEE = 'STEEN01'
     AND "NAME" = 'KDB01'
     AND TIMESTAMP = X'C5D4D8C3F3E7F3E5E8D2F5D6'
     AND DATEGRANTED = '230601'
     AND TIMEGRANTED = '15544316'
     AND GRANTEETYPE = ' '
     AND AUTHHOWGOT = ' '
     AND CREATETABAUTH = 'G' ……..
```

  - It works very well
  - Using a log tool to generate REDO DML illustrates PURE UPDATES
  - SYSVIEWS handled the same way – but SYSTABLES and SYSTABAUTH have an INSERT and DELETE

Security Cleanup Concerns Remediation (3|3)

- Db2 REVOKE_DEP_PRIVILEGES subsystem parameter
- The REVOKE_DEP_PRIVILEGES subsystem parameter controls whether revoking a privilege from a user is to cause dependent privileges to be revoked. If dependent privileges are to be revoked, revoking a privilege from a user also revokes the privilege from anyone that the user has granted that privilege to.
  - **NO** – REVOKE statements do not include dependent privileges. An error occurs if a REVOKE statement contains the INCLUDING DEPENDENT PRIVILEGES clause.
  - **YES** – REVOKE statements include dependent privileges, except when ACCESSCTRL, DATAACCESS, and system DBADM authorities are revoked. An error occurs if a REVOKE statement contains the NOT INCLUDING DEPENDENT PRIVILEGES clause, except when ACCESSCTRL, DATAACCESS, and system DBADM authorities are revoked.
  - **SQLSTMT** – Allows revoking of dependent privileges to be controlled at the SQL level, as specified in REVOKE statements. Db2 recognizes the dependent privileges clause (INCLUDING DEPENDENT PRIVILEGES or NOT INCLUDING DEPENDENT PRIVILEGES) of the REVOKE statement
- Note: This is a security-related parameter. If it is set to NO, privileges that were granted by a user are retained even if that user loses the authority that allowed the user to perform the grant

IDUG

2026

Sydney | March 16 - 18

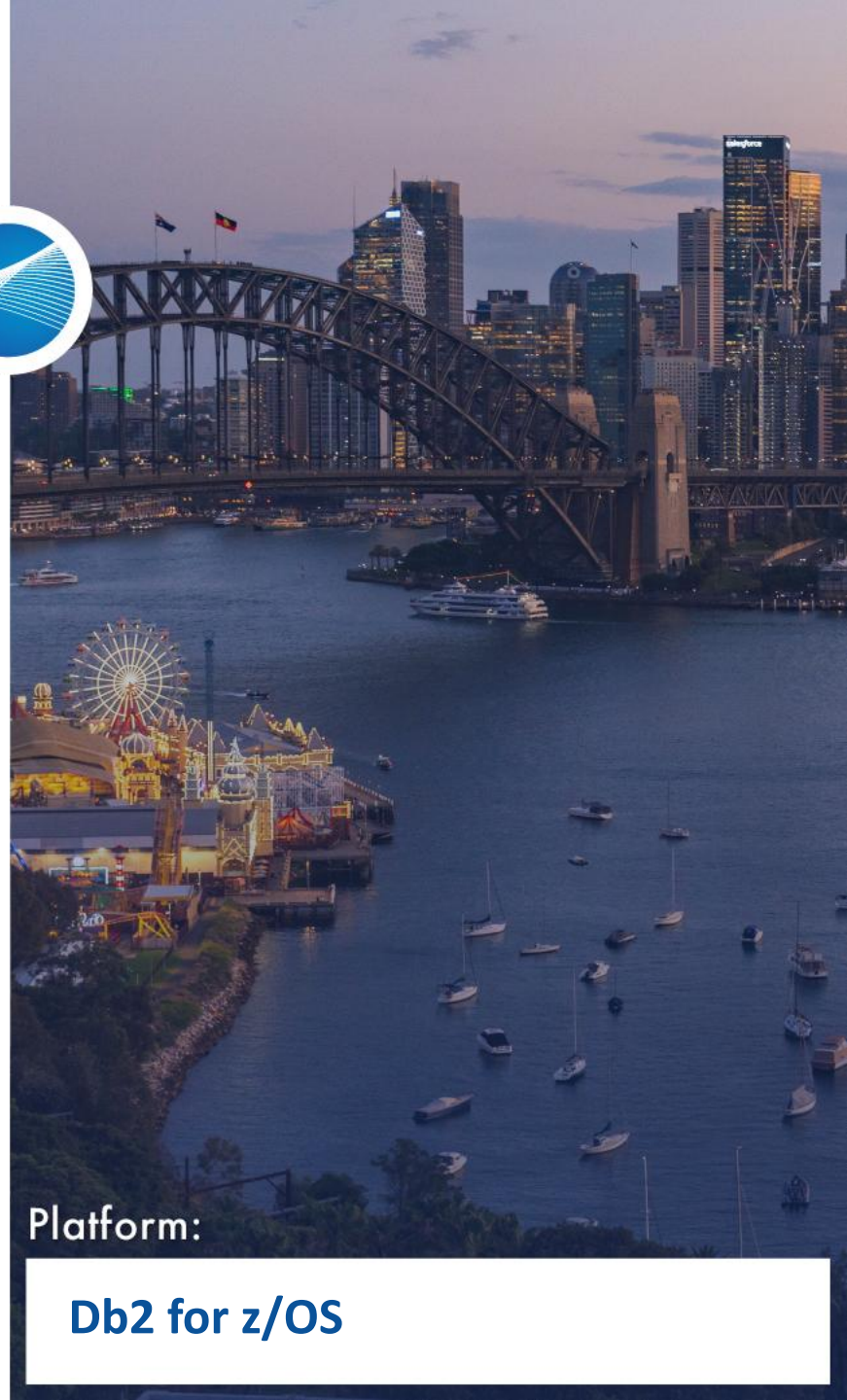
AU Db2 TECH CONFERENCE

Autonomous Auditing of Db2 Security Related Tables

Steen Rasmussen, *Broadcom*

Contact: steen.rasmussen@broadcom.com / db2steen@yahoo.com

Session Code: A11



Platform:

Db2 for z/OS

Thank You