COMMON EUROPE CONGRESS 2026

14 - 17 June
Lyon, France

The largest conference in Europe for solutions around IBM Power (IBM i, AIX, Linux) & IBM Storage

Session: RCAC - Row and Column Access Control
Secure your Data!

For: Common Europe Congress 2026
Date: 06/15/2026 05:05 pm

Birgitta Hauser
Modernization – Education – Consulting on IBM i
Diplom-Betriebswirt (BA)
Software and Database Architect
IBM Champion since 2020
eMail: Hauser@ModEdCon.com / Hauser@SSS-Software.de
Website: https://modedcon.com

IBM Champion since 2020

1

Landsberg am Lech – Stadtmauer / Townwall

IBM Champion since 2020

2

Agenda

Row And Column Access Control – Overview

- Roles and Separation of Duties
- Special Registers for RCAC
- Scalar Functions for RCAC

Implement and Activate Row Permissions

Implement and Activate Column Masks

- Update Rows with Column Masks

Restrictions and Pitfalls

3

Row and Column Access Control (RCAC)

4

What is RCAC?

RCAC = Row and Column Access Control

- **Additional layer** of **Data Security** (available with Db2)
  - CL commands for securing (database) objects can be used additionally
- **Complementary** to **Object Level Security**
- **Limits access** to **only** the **required data**
  - Controls access to a table at the **row** and/or **column level**
  - ***ALLOBJ users** can **no longer freely access all** of the **data** in the database

Provides 2 different approaches

- Access permissions for **rows** → `CREATE PERMISSION`
- Provides masks for **column contents** → `CREATE MASK`

IBM Advanced Data Security feature for i

- **Must be installed** → No-charge feature, option 47
- Required on **both development** and **production systems**

5

Why to use RCAC?

Current methods for limiting data access

- By defining and using **SQL Views**
- Access rules are included within the **application logic** → programming required

Access restrictions can be circumvented

- By accessing the database **table directly**
  - With interactive SQL, Db2-WebQuery, Query/400, JDBC, ODBC native I/O, UPDDTA etc.
- Users with **object authority** (e.g. *ALLOBJ) still can view **ALL data**

With RCAC ALL data access can be controlled at Column/Row Level

- **Indepenent** of which **access method** is used i.e. SQL, native I/O, CL, ODBC
- **No dependency** on **application logic** → **Business logic is moved into the database**
- Faciliates **multi-tenancy**
  - several independent customers/business units can **share a single database table** without being aware of one another
  - ensures each user **only sees the rows and column values** they are **truly entitled** to view

6

Roles and Separation of Duties

7

Roles and Separation of Duties

Current Situation

- **Data access roles** are defined in a **binary way,** i.e. **all or nothing**
  - **Access to ALL data** within a table (= object authority) or
  - **NO access to ANY data** within a table (= no object authority)
- **FULL access authority:** **All user profiles** accociated with the ***SECOFR user class**
  All user profiles with the ***ALLOBJ special authority**
  → **NO EXCEPTION**

Unfortunately, this might **not meet** the organization's **requirements for limiting access** to data or **separation of duties**

8

Roles and Separation of Duties

Function Usage IDs

- Allow granular security controls to be implemented
- Without granting users powerful special authorities

Function Usage ID Administration CL-Commands

- **WRKFCNUSG** Work with Function Usage
- **CHGFCNUSG** Change Function Usage
- **DSPFCNUSG** Display Function Usage

New Function Usage Ids

- **QIBM_DB_SECADM** Security Administrator function
- **QIBM_DB_SQLADM** Database Administrator function
- **QIBM_DB_SYSMON** Database Information function
- **QIBM_DB_ZDA** Toolbox application server access
- **QIBM_DB_DDMDRDA** DDM and DRDA® application server access

Roles and Separation of Duties - WRKFNCUSG CL Command

```
                          Work with Function Usage

Type options, press Enter.
  2=Change usage   5=Display usage

Opt  Function ID                    Function Name
 _   QIBM_DIRSRV_ADMIN              IBM Tivoli Directory Server Administrator
 _   QIBM_ACCESS_ALLOBJ_JOBLOG      Access job log of *ALLOBJ job
 _   QIBM_ALLOBJ_TRACE_ANY_USER     Trace any user
 _   QIBM_WATCH_ANY_JOB            Watch any job
 _   QIBM_DB_DDMDRDA               DDM & DRDA Application Server Access
 _   QIBM_DB_SECADM                Database Security Administrator
 _   QIBM_DB_SQLADM                Database Administrator
 _   QIBM_DB_SYSMON                Database Information
 _   QIBM_DB_ZDA                   Toolbox Application Server Access
 _   QIBM_QYAS_SERVICE_DISKMGMT     Disk units
 _   QIBM_SERVICE_DISK_WATCHER     DISK WATCHER
 _   QIBM_SERVICE_DUMP             Service dump
                                                                    More...
Parameters for option 2 or command
===> _____________________________________________________________________
F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel    F17=Top
F18=Bottom
```

- **New Database Function Usage Ids**

Services for Function Usage

FUNCTION_INFO View

- Details about the Function Usage Identifiers

```
Select Function_Id, Trim(Function_Name_Message_Text) Function_Name_Message_Text,
       Function_Type, Default_Usage, Function_Group_Id
  from Function_INFO a
  Where Function_Id like 'QIBM_DB%';
```

FUNCTION_ID	FUNCTION_NAME_MESSAGE_TEXT	FUNCTION_TYPE	DEFAULT_USAGE	FUNCTION_GROUP_ID
QIBM_DB	Database	GROUP	<NULL>	*NONE
QIBM_DB_SQLADM	Database Administrator	ADMINISTRABLE	DENIED	QIBM_DB
QIBM_DB_SYSMON	Database Information	ADMINISTRABLE	ALLOWED	QIBM_DB
QIBM_DB_SECADM	Database Security Administrator	ADMINISTRABLE	DENIED	QIBM_DB
QIBM_DB_DDMDRDA	DDM & DRDA Application Server Access	ADMINISTRABLE	ALLOWED	QIBM_DB
QIBM_DB_ZDA	Toolbox Application Server Access	ADMINISTRABLE	ALLOWED	QIBM_DB
QIBM_DB_GENCOL_OVERRIDE	Override Database Generated Values	ADMINISTRABLE	DENIED	QIBM_DB
QIBM_DB2_MIRROR	Db2 Mirror Administrator	ADMINISTRABLE	DENIED	QIBM_DB

FUNCTION_USAGE View

- Contains the Function Usage Configuration Details

```
Select * from Function_Usage
Where Function_Id like 'QIBM_DB_SECADM%';
```

FUNCTION_ID	USER_NAME	USAGE	USER_TYPE
QIBM_DB_SECADM	RPGPGM	ALLOWED	USER
QIBM_DB_SECADM	HAUSERPER	ALLOWED	USER

SQL_CHECK_FUNCTION_USAGE() User Defined Function

- Checks whether the effective User is authorized to the specified Function Usage Identifier
  - 0=User is not authorized / 1=User is authorized

```
Values(qsys2.SQL_Check_Function_Usage('QIBM_DB_SECADM'));
```

00001
0

Roles and Separation of Duties QIBM_DB_SECADM

```
CHGFCNUSG FCNID(QIBM_DB_SECADM)
          USER(DBSECOFR)
          USAGE(*ALLOWED)
```

- **Register User Profile DBSECOFR to be able to maintain data access abilities**

Only QSECOFR or a user with *SECADM authority

- can **grant** QIBM_DB_SECADM function usage to a user or group

User Profile registered for QIBM_DB_SECADM can

- Grant / revoke authority, change ownership, change primary group
- Grant read access to Db2 tables to other users

→ Users (even) with ***ALLOBJ authority** can be **prevented** from **reading data** in specific tables

- **The registered user profile itself is NOT able to read the content from any Db2 table**

Display Function Usage

```
                        Display Function Usage

Function ID  . . . . . :   QIBM_DB_SECADM
Function name  . . . . :   Database Security Administrator

Description  . . . . . :   Database Security Administrator Functions


Product  . . . . . :   QIBM_BASE_OPERATING_SYSTEM
Group  . . . . . :   QIBM_DB

Default authority  . . . . . :   *DENIED
*ALLOBJ special authority  . . . . . :   *NOTUSED

User        Type          Usage          User        Type          Usage
HAUSERPER   User          *ALLOWED
RPGPGM      User          *ALLOWED

F3=Exit    F12=Cancel    F17=Top    F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 2021.
```

- User **HAUSERPER** and **RPGPGM** are registered for **QIBM_DB_SECADM** function usage

```
Select * from Function_Usage
Where Function_Id like 'QIBM_DB_SECADM%';
```

FUNCTION_ID	USER_NAME	USAGE	USER_TYPE
QIBM_DB_SECADM	RPGPGM	ALLOWED	USER
QIBM_DB_SECADM	HAUSERPER	ALLOWED	USER

13

User Profile registered for QIBM_DB_SECADM Function

```
                    Change User Profile (CHGUSRPRF)

Type choices, press Enter.

User profile . . . . . > HAUSERPER     Name
User password  . . . . .   *SAME       Character value, *SAME, *NONE
Set password to expired  .   *NO        *SAME, *NO, *YES
Status . . . . . .   *ENABLED    *SAME, *ENABLED, *DISABLED
User class . . . . . .   *USER       *SAME, *USER, *SYSOPR...
Assistance level . . . . . .   *SYSVAL     *SAME, *SYSVAL, *BASIC...
Current library  . . . . . .   *CRTDFT     Name, *SAME, *CRTDFT
Initial program to call  . . .   *NONE      Name, *SAME, *NONE
  Library  . . . . . .                     Name, *LIBL, *CURLIB
Initial menu . . . . . .   MAIN        Name, *SAME, *SIGNOFF
  Library  . . . . . .     *LIBL      Name, *LIBL, *CURLIB
Limit capabilities . . . . . .   *NO         *SAME, *NO, *PARTIAL, *YES
Text 'description' . . . . . .   'Birgitta Hauser - FncUsr RCAC'

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this d
F24=More keys
```

- **User Profile HAUSERPER – Registered for QIBM_DB_SECADM**
  - **User Class: *USER**
  - **Without any special authority**

```
Select Authorization_Name USRPRF, Text_Description TEXT,
       User_Class_Name    USRCLS,
       Coalesce(Special_Authorities, '*NONE')      SPCAUT,
       Group_Profile_Name GRPPRF, Group_Authority  GRPAUT
  from User_Info_Basic a
  Where Authorization_Name = 'HAUSERPER';
```

USRPRF	TEXT	USRCLS	SPCAUT	GRPPRF	GRPAUT
HAUSERPER	Birgitta Hauser - FncUsr RCAC	*USER	*NONE	*NONE	*NONE

```
                    Change User Profile (CHGUSRPRF)

Type choices, press Enter.

                         Additional Parameters

Special authority  . . . . . .   *NONE       *SAME, *USRCLS, *NONE...
                + for more values
Special environment  . . . . .   *SYSVAL     *SAME, *SYSVAL, *NONE, *S36
Display sign-on information  . .   *SYSVAL     *SAME, *NO, *YES, *SYSVAL
```


14

Special Registers and scalar Functions

15

Special Registers important for RCAC

Special Registers	Description	Definition
`User`	Runtime user profile that determines the object authorities for the current connection/job.	`VARCHAR(18)`
`Session_User`		`VARCHAR(128)`
`Current_User`	Runtime user profile that determines the object authorities for the current connection/job.	
	Also reports adopted authority	
	Program/SQL routine created with **`USER=*OWNER`**	
	`Current_User` returns the **`*OWNER`** profile at runtime	
`System_User`	User Profile that initiated the connection to the server	
	prestarted jobs, initially connect to the server with a default user profile and then change to a different user profile	
	e.g. **`QUSER`** for a **`QZDASOINIT`** job	

16

VERIFY_GROUP_FOR_USER Scalar Function

```
VERIFY_GROUP_FOR_USER (Current User,
                       User/Group Profiles)
```

Checks the current user for access authority

- Primarily intended for use with **RCAC permissions** and **masks** but can also be used in **other SQL Statements**

Parameters

- ***Current User:*** `SESSION_USER, USER` or `CURRENT_USER` Special Registers
- ***User/Group Profiles:*** single or a list of IBM i User or Group Profiles

Returns an Integer-Value

- **1** = Special Register is **in** the specified list of users or group profiles
- **0** = Special Register is **not in** the specified list of users or group profiles

VERIFY_GROUP_FOR_USER Scalar Function - Example

```
Values(Current_User, Verify_Group_For_User(Current_User, 'QPGMR')),
      (Current_User, Verify_Group_For_User(Current_User, 'Hauser', 'MEIER',   'SCHMIDT')),
      (Current_User, Verify_Group_For_User(Current_User, 'Meier',  'QSYSOPR', 'Sales', 'HAUSERB')),
      (Current_User, Verify_Group_For_User(Current_User, 'ACCOUNT', 'HR', 'BHAPGMR', 'HAUSERHR'));
```

00001	00002
BHAPGMR	1
BHAPGMR	0
BHAPGMR	0
BHAPGMR	1

- **Current_User = 'BHAPGMR'**
- **User profile 'BHAPGMR' is a member of the group profile 'QPGMR'**

VERIFY_GROUP_FOR_USER Scalar Function - Example

```
Select s.*
  From Sales s
  Where CustNo = Case When Verify_Group_For_User(Current_User, 'HAUSERB') > 0
                      Then '10001'
                      When Verify_Group_For_User(Current_User, 'QPGMR') > 0
                      then '10002'
                      Else '10003' End
```

CUSTNO	ITEMNO	ITEM	SALESDATE	AMOUNT
10001	5100	King,Stephen - Es	2008-11-01	55,00
10001	5100	King,Stephen - Es	2008-12-23	60,00
10001	5200	King,Stephen - Drei	2009-01-30	160,00

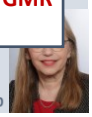
- Current_User = **HAUSERB** → Explicitly checked
- Returns only rows for **CUSTNO = 10001**

```
Select s.*
  From Sales s
  Where CustNo = Case When Verify_Group_For_User(Current_User, 'HAUSERB') > 0
                      Then '10001'
                      When Verify_Group_For_User(Current_User, 'QPGMR') > 0
                      then '10002'
                      Else '10003' End
```

- Current_User switched to **BHAPGMR**

CUSTNO	ITEMNO	ITEM	SALESDATE	AMOUNT
10002	5100	King,Stephen - Es	2008-11-15	1350,00
10002	5200	King,Stephen - Drei	2009-06-22	20,00
10002	5400	King,Stephen - Shining	2009-07-21	250,00

- Current_User = **BHAPGMR** → Member of **QPGMR**
- Returns only rows for **CUSTNO = 10002**



View: Check Current_User using VERIFY_GROUP_FOR_USER

```
Create or Replace View COMRCAC.SalesVRCAC
as Select s.*
      From Sales s
      Where CustNo = Case When Verify_Group_For_User(Current_User, 'HAUSERB')  > 0
                          Then '10001'
                          When Verify_Group_For_User(Current_User, 'QPGMR')    > 0
                          then '10002'
                          When Verify_Group_For_User(Current_User, 'HAUSER')   > 0
                          Then '10003'
                          Else '10004' End;
```

```
Select Current_User, CustNo, Sum(Amount) Total
   from SalesVRCAC
   Group By Current_User, CustNo;
```

00001	CUSTNO	TOTAL
HAUSERB	10001	3031,14

- Current_User: **HAUSERB**
- CUSTNO: **10001**

```
Select Current_User, CustNo, Sum(Amount) Total
   from SalesVRCAC
   Group By Current_User, CustNo;
```

00001	CUSTNO	TOTAL
BHAPGMR	10002	2986,25

- Current_User: **BHAPGMR**
- Group Profile: **QPGMR**
- CUSTNO: **10002**

VERIFY_GROUP_FOR_USER Example

```
Select Current_User CurrUser, EmployeeNo, LastName, CostCenter,
       Case When Verify_Group_For_User(Current_User, 'HAUSERB') > 0
            Then Case When CostCenter = 344              Then Salary Else 0 End
            When Verify_Group_For_User(Current_User, 'OPGMR') > 0
            Then Case When CostCenter Between 100 and 199 Then Salary Else 0 End
            When Verify_Group_For_User(Current_User, 'HAUSERHR') > 0
            Then Salary
            Else 0
            End DspSalary
  From comrcac.EmployeeCpy a;
```

CURRUSER	EMPLOYEENO	LASTNAME	COSTCENTER	DSPSALARY
HAUSERB	1000	Fischer	344	55000,00
HAUSERB	1010	Meier	344	105000,00
HAUSERB	1020	Bauer	344	55000,00
HAUSERB	2000	Schmidt	100	0,00
HAUSERB	2100	Gerber	111	0,00
HAUSERB	1030	Moser	344	30000,00

CURRUSER	EMPLOYEENO	LASTNAME	COSTCENTER	DSPSALARY
BHAPGMR	1000	Fischer	344	0,00
BHAPGMR	1010	Meier	344	0,00
BHAPGMR	1020	Bauer	344	0,00
BHAPGMR	2000	Schmidt	100	200000,00
BHAPGMR	2100	Gerber	111	45000,00
BHAPGMR	1030	Moser	344	0,00

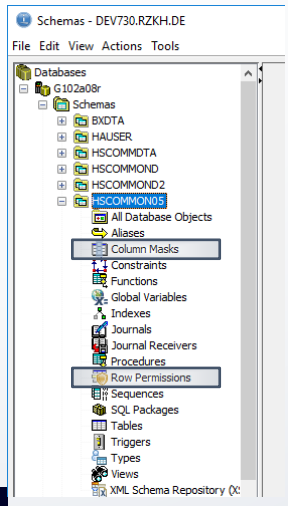
CURRUSER	EMPLOYEENO	LASTNAME	COSTCENTER	DSPSALARY
HAUSERHR	1000	Fischer	344	55000,00
HAUSERHR	1010	Meier	344	105000,00
HAUSERHR	1020	Bauer	344	55000,00
HAUSERHR	2000	Schmidt	100	200000,00
HAUSERHR	2100	Gerber	111	45000,00
HAUSERHR	1030	Moser	344	30000,00

CURRUSER	EMPLOYEENO	LASTNAME	COSTCENTER	DSPSALARY
HAUSERPER	1000	Fischer	344	0,00
HAUSERPER	1010	Meier	344	0,00
HAUSERPER	1020	Bauer	344	0,00
HAUSERPER	2000	Schmidt	100	0,00
HAUSERPER	2100	Gerber	111	0,00
HAUSERPER	1030	Moser	344	0,00

23

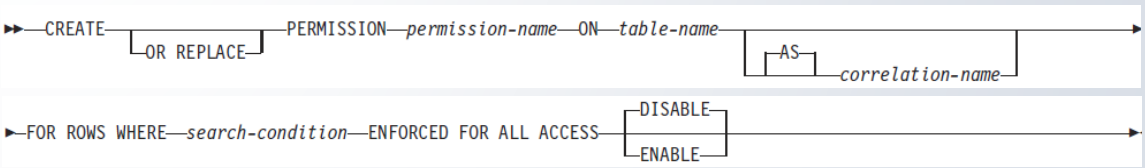
24

Access Client Solutions – Schema - Row and Column Access Control (RCAC)



- **Column Masks**
- **Row Permission**

Row Permission - Create Permission Statement



CREATE PERMISSION = Define Row Permission

- *Permission-Name:* **Name** of the RCAC **row permission**
- *Table-Name:* **Table** on which the row permission is created
- FOR ROWS: **Row** Access Control
- *Seach Condition:* Where Conditions that result in either **true, false** or **unknown**
- ENFORCED FOR ALL ACCESS → **Enable/Disable:** Permission initially enabled or not

Alter Table – Activate Row and Column Access Control

A Row Permission is NOT activated with creation

- Even though **ENFORCED FOR ALL ACCESS is enabled**

A Row Permission needs to be activated on the Table

- An **ALTER TABLE** statement must be performed for **explicitly activating** the **Row Access Control** (i.e. Row Permission)

```
ALTER TABLE HSCOMMON10.MYADDRESS
            ACTIVATE ROW ACCESS CONTROL ;
```


Data Access with Row Permission - Example

```
CREATE OR REPLACE PERMISSION HSCOMMON10.MYADDRESS_PERMCUSTNO ON HSCOMMON10.MYADDRESS AS MACUSTNO
    FOR ROWS
        WHERE Substr(CustNo, 1, 1)
                between case When Verify_Group_for_User(Session_User, 'HAUSERB') = 1 Then '0'
                             When Verify_Group_for_User(Session_User, 'QPGMR')   = 1 Then '5'
                        End
                    and case When Verify_Group_for_User(Session_User, 'HAUSERB') = 1 Then '4'
                             When Verify_Group_for_User(Session_User, 'QPGMR')   = 1 Then '7'
                        End
    ENFORCED FOR ALL ACCESS
    ENABLE  ;
```

- Session_User = **HAUSERB**
- Can access all customer no beginning with **0, 1, 2, 3** or **4**

```
Select * from HSCOMMON10.MyAddress
```

CUSTID	CUSTNO	CUSTNAME1
45	00100	Pallhuber und Söhne
46	00110	Bahnleitner Gemischtwaren
47	00120	Ebäcko Nordrhein Westfalen
48	00130	Deutscher-Paket-Dienst

- Session_User = **BHAPGMR**
- Member of the **QPGMR** group profile
- Can access all customer no beginning with **5, 6** or **7**

```
Select * from HSCOMMON10.MyAddress
```

CUSTID	CUSTNO	CUSTNAME1
31	56453	GWINNER WOHNDESIGN GMBH
23	63820	FIRMA MAYER GMBH
29	63899	HELLSTERN GMBH
33	66215	GÜNTHER NETZER GMBH
36	66588	ARBURG GMBH

Queries, Totals and Column Masks

Queries and Aggregate Functions with Column Maks

- Column Values depending on the User‘s access authorities

- **Mask Value = -9999!!!**

```
Select Session_User, Sum(Salary) Total
  from Employee;
```

00001	TOTAL
BHAPGMR	258002,00

```
Select Session_User, Sum(Salary) Total
  from Employee;
```

00001	TOTAL
HAUSERB	198002,00

```
Select Session_User, Sum(Salary) Total
  from Employee;
```

00001	TOTAL
HAUSER2	-5994,00

- Session_User = **BHAPGMR**
- Member of the QPGMR User Profile
- Only allowed for Salaries in Cost Center **344**

- Session_User = **HAUSERB**
- Allowed for all Salaries **< 100.000 Euro**

- Session_User = **HAUSER2**
- Is not allowed for any Salaries

Column Masks

Column Permission - Grant / Revoke Statements

GRANT Statement

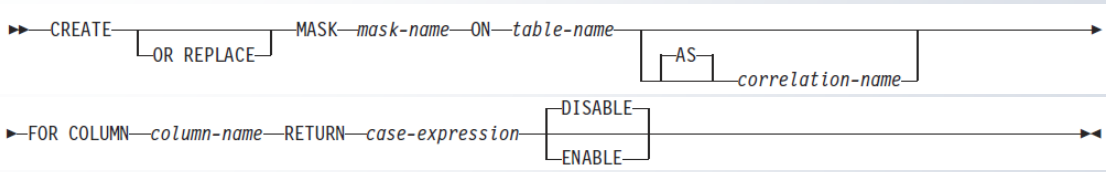
- grants **privileges** on **tables** or **views**
  to a specific **user**, **group profile** or to **PUBLIC** (special value)
- Grants the privilege to **update only those columns** that are **explicitly listed** within the GRANT statement.
  - Other columns within the table or view that are **not explicitly listed cannot be modified** by the specified user, group profile or PUBLIC

```
GRANT SELECT ,
      UPDATE ( BIRTHDAY   , CITY   , COSTCENTER , COUNTRY   ,
               DEPARTMENT , EMAIL  , EMPLOYEENO , EXITDATE  ,
               FIRSTNAME  , "ID"   , JOBSPEC    , LASTNAME  ,
               MOBILE     , PHONE  , STREET     , TITLE     ,
               ZIPCODE )
    ON HSCOMMON10.EMPLOYEE
    TO HAUSERB ;
```

- User Profile **HAUSERB** can
  - **Read** data from the **EMPLOYEE** table
  - **Update** the explicitly **specified columns** only

32

Column Permission – Column Masks - Create Mask Statement



CREATE MASK – Hide column contents

- *Mask-Name* **Name of the mask** for column access control
- *Table-Name* **Table** on which the column mask is created
- *Correlation-Name* optional correlation name that can be used within the case-expression
- FOR COLUMN *Column-Name* **Column** to which the mask applies
- RETURN *Case-Expression* **Case-Expression** to be evaluated
  Data types of the result/mask value and the based column must **be compatible**
- Enable/Disable is to be initially enabled or not

33

Alter Table - Activate Row and Column Access Control

A Column Mask is NOT activated with creation

- Even though **ENABLED** is specified

A Column Mask needs to get activated on the table

- An **ALTER TABLE** statement must be performed for **explicitly activating** the **Column Access Control** (i.e. Column Masks)

```
ALTER TABLE HSCOMMON10.EMPLOYEE
            ACTIVATE COLUMN ACCESS CONTROL ;
```


Data Access with Column Mask Example

```
CREATE or Replace MASK COMRCAC.PEREMPLSALARY
        ON COMRCAC.EMPLOYEE
        FOR COLUMN SALARY
  RETURN Case When Verify_Group_For_User(Session_User, 'HAUSERB')    = 1
              Then Case When salary < 100000 Then Salary Else -999 End
              When   Verify_Group_For_User(Session_User, 'QPGMR')    = 1
                 and CostCenter = 344        Then Salary
              When   Verify_Group_For_User(Session_User, 'HAUSERHR') = 1
              Then Salary Else -999 End
   ENABLE;
```

```
Select Id, EmployeeNo, CostCenter,
       Salary, LastName, FirstName
  from Employee
  Order By Salary Desc
```

ID	EMPLOYEENO	COSTCENTER	SALARY
4	2000	100	-999,00
2	1010	344	70000,00
1	1000	344	55000,00
3	1020	344	55000,00
5	2100	111	45000,00
6	1030	344	30000,00

- Session_User = **HAUSERB**
- Can see all salaries **lower than 100000**

```
Select Id, EmployeeNo, CostCenter,
       Salary, LastName, FirstName
  from Employee
  Order By Salary Desc
```

ID	EMPLOYEENO	COSTCENTER	SALARY
4	2000	100	-999,00
2	1010	344	70000,00
1	1000	344	55000,00
3	1020	344	55000,00
5	2100	111	-999,00
6	1030	344	30000,00

- Session_User = **BHAPGMR**
- Member of Group Profile **QPGMR**
- Sees only salaries of **cost center 344**

Employee Master and Credit Card No - Example

```
Select EmployeeNo, EMCCID "CredCard Id",
       LastName, FirstName, Street, ZipCode, City
from EmplCred
Order By EMCCID;
```

- Employee Master

EMPLOYEENO	CredCard Id	LASTNAME	FIRSTNAME	STREET	ZIPCODE	CITY
2100	2	Gerber	Kim	Am Bach 3	85051	Ingolstadt
1000	5	Fischer	Fritz	Oberfeldstr. 16	76149	Karlsruhe
1010	11	Meier	Anna	Frankfurter Str. 55	63128	Dietzenbach
1020	15	Bauer	Stefan	An der Havel 234	10785	Berlin
1030	17	Moser	Ben	Waldstr 1	77880	Sasbach
2000	19	Schmidt	Anton	Seestr. 7	17192	Waren/Mueritz

```
Select CCID "CredCard Id", CredCardNo, CVV,
       ExpirYear, ExpirMonth
  from CREDCARD;
```

- Credit Card Table

CardId	CREDCARDNO	CVV	EXPIRYEAR	EXPIRMONTH
1	6011276617038831	765	2022	3
2	5140103430432676	221	2022	9
3	5294479349873539	396	2021	5
4	5302985801465861	324	2022	9
5	5487796361329240	323	2024	9
6	4013987007326163	373	2021	1
7	4042051746869564	614	2021	7
8	4047043193104590	207	2021	2
9	4070199796656947	793	2021	12
10	4092983562516516	460	2022	7
11	4108554684486359	210	2024	11
12	4131077380803421	833	2022	10
13	4134875449544952	376	2021	10
14	4500484439016885	602	2022	3
15	4502245604298774	263	2024	3
16	4508969691406214	586	2021	7
17	4529749847757685	127	2023	3
18	4539759236009026	716	2021	5
19	4563063840841158	503	2024	11

- Employee Table with Id of the Credit Card Table
- Whether the Credit Card No is (partly)masked or not depends on information from the Employee Table (Cost Center)

Employee Master and Credit Card No - Example

```
Create Or Replace Mask COMRCAC.COLM_CREDCARD_CREDCARDNO
    On COMRCAC.CREDCARD For Column CREDCARDNO
       Return Case When     Verify_Group_For_User(Session_User, 'HAUSERHR') = 1
                    Then CredCardNo
                    When     Verify_Group_For_User(Session_User, 'HAUSERB')  = 1
                         and (Select CostCenter
                                 From EmplCred
                                 Where EMCCID = CCID
                                 Fetch First Row Only) between 100 and 200
                    Then CredCardNo
               When     Verify_Group_For_User(Session_User, 'QPGMR') = 1
                    and (Select CostCenter
                          from EmplCred
                          Where emccid = CCID
                                Fetch First Row Only) = 344
                    Then CredCardNo
                    Else Repeat('*', Length(Trim(CredCardNo)) - 4) concat Right(Trim(CredCardno), 4)
               End

        Enable;
```

- Cost Center from the Employee Master Table is checked
- The last 4 digits of the credit card no are always visible

```
ALTER TABLE COMRCAC.CREDCARD
            ACTIVATE COLUMN ACCESS CONTROL ;
```

- Activate Column Access Control

Credit Card Table with Column Masks - Example

```
Select CCID "CredCard Id", CredCardNo, CVV,
       ExpirYear, ExpirMonth
   from CREDCARD;
```

CredCard Id	CREDCARDNO	CVV	EXPIRYEAR	EXPIRMONTH
1	**********8831	765	2022	3
2	**********2676	221	2022	9
3	**********3539	396	2021	5
4	**********5861	324	2022	9
5	5487796361329240	323	2024	9
6	**********6163	373	2021	1
7	**********9564	614	2021	7
8	**********4590	207	2021	2
9	**********6947	793	2021	12
10	**********6516	460	2022	7
11	4108554684486359	210	2024	11
12	**********3421	833	2022	10
13	**********4952	376	2021	10
14	**********6885	602	2022	3
15	4502245604298774	263	2024	3
16	**********6214	586	2021	7
17	4529749847757685	127	2023	3
18	**********9026	716	2021	5
19	**********1158	503	2024	11

Update with Column Masks

What happens if a complete row is updated but the row buffer inlcudes mask values?

- Example: The postal address has to be updated in the employee master table.
  The birthday is a different column within the same table.
  A **column mask** is added to the **birthday** column.
  The user who has to update the postal address is not allowed to see the birthday, instead he **sees the mask**.
  When updating the row with native I/O the **record buffer** does **not** include the **original**, but the **masked** birthday.

How to preserve the original value?

- Adding a **check constraint** with the **ON VIOLATION** clause
- Adding a **SECURED before insert/update trigger**

Check Constraint with ON VIOLATION Clause

```
Alter Table EMPLOYEE
      Add Constraint ChkCst_Employee_Birthday
          Check(BIRTHDAY > '0001-01-01')
          On Insert Violation Set      BIRTHDAY = Default
          On Update Violation Preserve BIRTHDAY;
```

- **ON INSERT VIOLATION**
- **If a column mask is passed at insert it is replaced with the default value**

- **ON UPDATE VIOLATION**
- **If a column mask is passed at update, the previous value is preserved**

SECURED Attribute

SECURED is needed for Views, Functions, Triggers

- A **SECURED** trigger or function is **considered secure for RCAC**
  - Must be specified for a **trigger** based on **table with RCAC**
  - Must be specified for an **instead of trigger** that is created for a view where one or more of the **underlying tables is using RCAC**
  - Must be specified when **a function** is referenced in a **RCAC mask**

Before Trigger with the SECURED Attribute

```
Create or Replace Trigger HSCOMMON10.REPLACE_MASK_BIRTHDAY
       Before Insert Or Update On HSCOMMON10.EMPLOYEE
       Referencing New Row as N
                   Old Row as O
       For Each Row
       Mode DB2ROW
       Secured
       When (N.BIRTHDAY = '0001-01-01')
   Begin
     If     Inserting Then Set N.BIRTHDAY = Default;
     ElseIf Updating  Then Set N.BIRTHDAY = o.BIRTHDAY;
     End If;
   End;
```

- SECURED Before Insert/Update Trigger for preserving original values
- If a column mask is passed at insert it is replaced with the default value
- If a column mask is passed at update, the previous value is preserved

RCAC and Native I/O – Example - Without Column Mask

DSPLY	Programming	344	Bauer	Stefan	70000.00
DSPLY	Programming	344	Fischer	Fritz	55000.00
DSPLY	Programming	344	Meier	Anna	105000.00
DSPLY	Programming	344	Moser	Ben	30000.00
DSPLY	Programming	344			260000.00
DSPLY	Programming				260000.00
DSPLY	Sales	100	Schmidt	Anton	150000.00
DSPLY	Sales	100			150000.00
DSPLY	Sales	111	Gerber	Kim	45000.00
DSPLY	Sales	111			45000.00
DSPLY					455000.00

- Program Salary02:
  - Accumulate the annual salary per Cost Center and Department

Add Column Masks for Salary and Birthday

```
CREATE or Replace MASK COMRCAC.COLM_Employee_Salary
         ON COMRCAC.EMPLOYEE
         FOR COLUMN SALARY
  RETURN Case When Verify_Group_For_User(Session_User, 'HAUSERHR')    = 1
              Then Salary
              When Verify_Group_For_User(Session_User, 'HAUSERB')     = 1
              Then Case When salary < 100000 Then Salary Else -999 End
              When   Verify_Group_For_User(Session_User, 'QPGMR')     = 1
                   and CostCenter = 344       Then Salary
             Else -999 End
   ENABLE;
```

- **Attention:** When setting numeric Default Values to anything other than *Zeros!!!

```
Create Or Replace Mask COMRCAC.COLM_EMPLOYEE_BIRTHDAY
    On COMRCAC.EMPLOYEE For Column BIRTHDAY
       Return Case When     Verify_Group_For_User(Session_User, 'QPGMR') = 1
                        And COSTCENTER = 344
                    Then BIRTHDAY
                    When Verify_Group_For_User(Session_User, 'HAUSERHR') = 1
                    Then BIRTHDAY
                    Else '0001-01-01' End
        Enable;
```


RCAC and Native I/O – With Column Mask Example

DSPLY	Programming	344	Bauer	Stefan	70000.00	70000.00	70000.00
DSPLY	Programming	344	Fischer	Fritz	55000.00	55000.00	55000.00
DSPLY	Programming	344	Meier	Anna	105000.00	-999.00	105000.00
DSPLY	Programming	344	Moser	Ben	30000.00	30000.00	30000.00
DSPLY	Programming	344			260000.00	154001.00	260000.00
DSPLY	Programming				260000.00	154001.00	260000.00
DSPLY	Sales	100	Schmidt	Anton	-999.00	-999.00	150000.00
DSPLY	Sales	100			-999.00	-999.00	150000.00
DSPLY	Sales	111	Gerber	Kim	-999.00	45000.00	45000.00
DSPLY	Sales	111			-999.00	45000.00	45000.00
DSPLY					258002.00	198002.00	455000.00

- Run the Program with different users
- BHAPGMR: Mask the Salaries for all employess not in cost center 344
- HAUSERB: Mask all Salaries higher than 100,000
- HAUSERHR: See all Salaries

- Different Totals and Grand Total

Add Check Constraints for Birthday and Salary

```
Alter Table EMPLOYEE
      Add Constraint ChkCst_Employee_Birthday
          Check(BIRTHDAY > '0001-01-01')
          On Insert Violation Set       BIRTHDAY = Default
          On Update Violation Preserve BIRTHDAY;
```

```
Alter Table EMPLOYEE
     Add Constraint ChkCst_Employee_Salary
         Check(Salary >= 0)
         On Insert Violation Set       Salary = Default
         On Update Violation Preserve Salary;
```

- Without Check constraint with ON INSERT and ON UPDATE violation or an appropriate BEFORE INSERT OR UPDATE Trigger, the **masked values are written**!!!

RCAC and Native I/O – With Column Mask Update Example

- Update Recored with Native I/O (includes all columns)

```
DSPLY  User: HAUSER
DSPLY  Fischer Fritz 86916 Kaufering
DSPLY  CostCenter:344 Birthday: 1958-05-15 Salary: 55000.0
DSPLY  Fischer Fritz 76149 Karlsruhe
DSPLY  CostCenter:344 Birthday: 1958-05-15 Salary: 55000.0
DSPLY  Schmidt Anton 63303 Langen
DSPLY  CostCenter:100 Birthday: 0001-01-01 Salary: -999.00
DSPLY  Schmidt Anton 17192 Waren/Mueritz
DSPLY  CostCenter:100 Birthday: 0001-01-01 Salary: -999.00
```

- **BHAPGMR:**
  Is only allowed to see birthdays and salaries from employees in Cost center 344

```
DSPLY  User: HAUSERHR
DSPLY  Fischer Fritz 86916 Kaufering
DSPLY  CostCenter:344 Birthday: 1958-05-15 Salary: 55000.0
DSPLY  Fischer Fritz 76149 Karlsruhe
DSPLY  CostCenter:344 Birthday: 1958-05-15 Salary: 55000.0
DSPLY  Schmidt Anton 63303 Langen
DSPLY  CostCenter:100 Birthday: 1965-01-31 Salary: 150000.
DSPLY  Schmidt Anton 17192 Waren/Mueritz
DSPLY  CostCenter:100 Birthday: 1965-01-31 Salary: 150000.
```

- **HAUSERHR:**
  Can see and manage birthdays and salaries from all employees

Restrictions

Files that cannot be handled with RCAC active

- Distribued files
- Internally/program described files
- Multi-format logical files
- Files with ICU 2.6.1 Sort Sequence
- Files or Tables with READ triggers

Data Access must be performed with the SQE

- Data access with **non-SQL interfaces** is now **executed with the SQE**
  - Non-SQL Interfaces: Native I/O (RPG or Cobol), Query400, OPNQRYF, RUNQRY, QQQQRY API
- **RCAC cannot be used**
  - When the data access is **rerouted to** by setting the SQE_NATIVE_ACCESS Option in the QAQQINI file to *NO
  - Beginning With **Release 7.4 SQE_NATIVE_ACCESS** Option will **be ignored**

RCAC active – Copy Data with the CPYF – CL Command

Copy Data with the CPYF (Copy File) CL Command

- Only data the **user can access** is copied
  - **Not all rows** may be copied
  - If **column masks** are included the **masked values** are copied!
- When a **new table** is created, **RCAC definitions** are **not copied**
- With **different RCAC definitions** in the From and To File the CPYF command may crash

- **Attention:** Incorrect RCAC implementation may cause **program crashes** or even worse **data may get lost** without any error message because incompletely copied

Additional Consideration

Data Transfer from a Table with RCAC into a Table without RCAC

- Only the rows the **user is allowed to access** are transferred
- **Masked** values **are transferred**

• **Potential data loss**

Data Transfer into a Table with RCAC from a Table without RCAC

- Only data that the **user is able to read** can be transferred

• **Data may be incompletely copied**

Different RCAC Definition in both Tables

- Only rows that the **user is able to read** can be inserted into the new table
- **Masked** values are **denied**

• **Transfer may crash**

RCAC active – Create duplicate Object with the CRTDUPOBJ CL-Command

Create Duplicate Object with the CRTDUPOBJ command

- **New Option:** AC (Duplicate access control)
  - ***ALL** All RCAC definitions are copied to the new table (= Default)
  - ***ROW** Only Row-Permissions are copied
  - ***COL** Only Column masks are copied
  - ***NONE** RCAC definitions are NOT copied
- Option **DATA = *YES**
  - **All data (**all rows and column values) is copied
  - **DATA = *YES** → **ACCCTL** must be set to ***ALL** otherwise the duplicate cannot be created
- **Attention:** With **CPYF** only **data the user can access** is copied
  With **CRTDUPOBJ** the **complete object** with **all data** is copied

CRTDUPOBJ – Create duplicate Object - Enhancement

```
                    Create Duplicate Object (CRTDUPOBJ)

Type choices, press Enter.

From object  . . . . . > EMPLOYEE     Name, generic*, *ALL
From library . . . . . > COMRCAC      Name, *LIBL, *CURLIB
Object type  . . . . . > *FILE        *ALL, *ALRTBL, *AUTL...
             + for more values
To library . . . . . .   *FROMLIB     Name, *FROMLIB, *SAME...
New object . . . . . .   *OBJ         Name, *OBJ, *SAME
From ASP device  . . . .   *          Name, *, *CURASPGRP, *SYSBAS
To ASP device  . . . . .   *ASPDEV     Name, *ASPDEV, *...
Duplicate data . . . . .   *Yes        *NO, *YES
Duplicate constraints  .   *YES       *YES, *NO
Duplicate triggers . . .   *YES       *YES, *NO
Duplicate file identifiers .   *NO    *NO, *YES
Duplicate access control . .   *ALL    *ALL, *ROW, *COL, *NONE

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

- If Duplicate Data is set to *YES
  Duplicate Access Control must be set to *ALL

Additional Considerations

RCAC Timing

- column masking occurs **after ALL Query Processing** is complete
- Local Selection, Joins, Group Bys and Sorts **based** on the **masked values**

RCAC and Field Procedures

- Field Procedure Masking when **reading/writing column values**

  → **Before Query Processing** # RCAC column masking

RCAC and Data Movement

- For data movement processes (copy/save) **permission to access ALL data without any masking**

  → **critical! Must be designed and planned carefully**

RCAC and Journal Receivers

- RCAC is **not applied** to **Journal Receiver Access,**

  → i.e. **all transactions independent of RCAC** are found in the journal receivers

Any Questions?

Special Thanks to

Holger Scherer – RZKH Rechenzentrum Kreuznach

- For providing an IBM i-System enabling the creation of the samples/code used in my presentations
- http://www.rzkh.de

RZKH RECHENZENTRUM KREUZNACH

Your data is save! … in the bunker

Speaker‘s Biography

Birgitta Hauser
Diplom-Betriebswirt (BA)
Database and Software Architect

Birgitta Hauser worked on the IBM i and its predecessors since 1992. She graduated with a business economics diploma, and started programming on the AS/400 in 1992. She worked and works as traditional RPG Programmer but also as Database and Software Engineer, focusing on IBM i application and database modernization.

Currently she is self-employed and works in Consulting and Application and Database Modernization on IBM i and Db2 for i. Since July, 2019 she is occasionally working for Fresche Solutions Inc. (Montréal) as a contractor.

She also works in education as a trainer for RPG and SQL developers.

Since 2002 she has frequently spoken at the COMMON User Groups and other IBM i and Power Conferences in Germany, other European Countries, USA and Canada.

In addition, she is co-author of two IBM Redbooks and also the author of several articles and papers focusing on RPG and SQL for the ITP Verlag (a German publisher), IT Jungle Guru and IBM DeveloperWorks.

In 2015 she received the John Earl Speaker Scholarship Award. In 2018 she received the Al Barsa Memorial Scholarship Award.

IBM Champion since 2020

58

COMMON EUROPE CONGRESS 2026
14 - 17 June
Lyon, France
The largest conference in Europe for solutions around IBM Power (IBM i, AIX, Linux) & IBM St

Thank you!

Secure your Data with
Row and Column Access Control (RCAC)?
Yes i can!

If you are interested in more detailed individual Workshops on-site or remote, Please contact me directly

Birgitta Hauser – Modernization – Education – Consulting on IBM i
Diplom-Betriebswirt (BA)
Database and Software Architetct
IBM Champion since 2020
eMail: Hauser@ModEdCon.com / Hauser@SSS-Software.de
Web: https://modedcon.com/

IBM Champion since 2020

59