RCAC

Hints & Tips

By Mark Gillis

RCAC: Usage Considerations

Part 1

Row and Column Access Control (RCAC) in DB2 10.1 for LUW is a neat and simple means of shielding your data from users who don't need to see it. It's really two things melded together:

- Row **permissions** : which only return the rows of data that you are permitted to see
- Column **masks** : which over-write or obscure returned values with specified alternatives

Used together they can provide a useful security layer but there are a few usage considerations to be aware of.

To set the scene, I have a table called LOGBOOK which contains details of a private pilot's flying log. A user query has been run to look at what was logged in the year 2011. The SQL for this query is shown below and this query will be used throughout :

```
Select LOGBOOK_KEY,
       Date(Brakes_Off) as FLIGHT_DATE,
       REGISTRATION,
       CAPACITY,
       PILOT,
       FROM_AIRFIELD,
       CHECK_PILOT_ID
       from LOGBOOK.logbook
       where year(Brakes_Off) = '2011'
       order by Brakes_Off
```

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        232 24.01.2011  G-AYJR       P1       Self                      EGBT          -
        233 17.04.2011  HB-CFN       P1S      Self                      LFSB          -
        234 30.04.2011  G-AYJR       P1S      Self                      EGBT          UKFE236606A/A
        235 21.06.2011  HB-CFN       P1S      Self                      LFSB          -
        236 21.06.2011  HB-CFN       P1S      Self                      LFGB          -
        237 23.07.2011  G-ETME       P1S      Self                      EGLM          -
        238 28.07.2011  HB-CFN       P1S      Self                      LFSB          -
        239 28.07.2011  HB-CFN       P1S      Self                      LSZQ          -
        240 11.10.2011  G-BORK       P1       Self                      EGBT          -
```

The LOGBOOK_KEY is a generated Primary Key, the airfields are shown using standard International Civil Aviation Organization (ICAO) codes and there is one row with a check pilot ID, indicating that the flight was a test of some sort.

Column Masks

The DBAs first task is to obfuscate the ICAO codes; specifically to change the value to UK where it's an airfield in the UK (as indicated by a leading letter E) and XXXX for anywhere else. The column mask definition for this is

```
CREATE or REPLACE MASK FOREIGN_AIRFIELD ON LogBook FOR
       COLUMN from_airfield RETURN
       CASE
              WHEN SUBSTR(from_airfield,1,1) = 'E' THEN 'UK'
              ELSE 'XXXX'
       END
ENABLE;
```

And it is activated with this statement

```
ALTER TABLE LogBook ACTIVATE COLUMN ACCESS CONTROL;
```

Now the user running the same flight details query will see the following results

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        232 24.01.2011  G-AYJR       P1       Self                      UK            -
        233 17.04.2011  HB-CFN       P1S      Self                      XXXX          -
        234 30.04.2011  G-AYJR       P1S      Self                      UK            UKFE236606A/A
        235 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        236 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        237 23.07.2011  G-ETME       P1S      Self                      UK            -
        238 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
        239 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
        240 11.10.2011  G-BORK       P1       Self                      UK            -
```

There are two things to be aware of here from the user's point of view:

1. The fact that you can't see the underlying data does not prevent you actioning it
2. Your SQL will have the column mask inserted into it by the optimizer

To illustrate the first point, the user could run this INSERT statement

```
Insert into Logbook
(REGISTRATION,CAPACITY,PILOT,FROM_AIRFIELD,TO_AIRFIELD,BRAKES_OFF,BRAKES_ON,Landings,COMMENTS)
Values
('G-AVLT','P1S','Self','LSZQ','LSZQ','2011-07-30-14.35.00.000000','2011-07-30-15.35.00.000000',6,'TEST
ENTRY')
```

which includes a non-UK ICAO code (LSZQ). That will work fine, but when the user re-runs the query, an extra row is visible but not the ICAO code that was just inserted (LOGBOOK_KEY 14).

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        232 24.01.2011  G-AYJR       P1       Self                      UK            -
        233 17.04.2011  HB-CFN       P1S      Self                      XXXX          -
        234 30.04.2011  G-AYJR       P1S      Self                      UK            UKFE236606A/A
        235 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        236 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        237 23.07.2011  G-ETME       P1S      Self                      UK            -
        238 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
        239 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
         14 30.07.2011  G-AVLT       P1S      Self                      XXXX          -
        240 11.10.2011  G-BORK       P1       Self                      UK            -
```

This is because the Column Mask was created without any context checking. *Anyone* accessing this data will now have the data obfuscated for them. In order to address this issue, the column mask definition needs to be changed to allow some users, maybe just the DBAs, to see the raw data. This can be accomplished by setting up a ROLE and assigning specific users to that ROLE, e.g.

```
CREATE ROLE DBA
GRANT ROLE DBA to USER userid
```

And including the following clause as the first WHEN clause in the Column Mask definition

```
      WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'DBA') = 1 THEN from_airfield
```

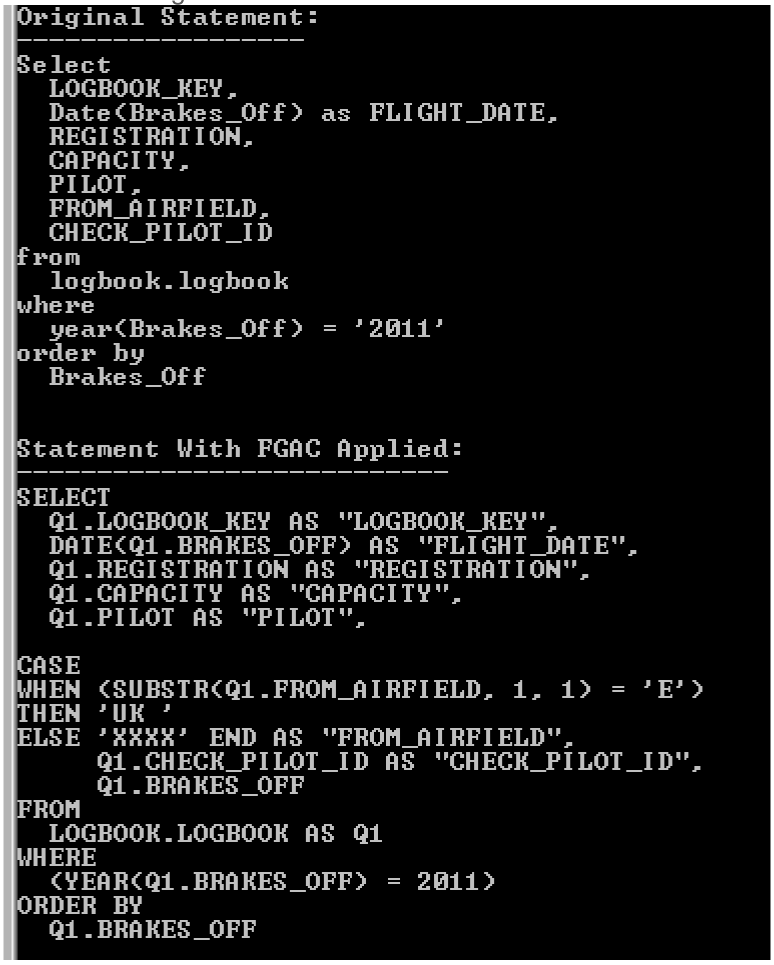
The contextual clauses in the Column Mask definition can be as complex or as simple as you like. For example, I can redefine the mask as:

```
CREATE or REPLACE MASK FOREIGN_AIRFIELD ON LogBook FOR
      COLUMN from_airfield RETURN
            CASE
            WHEN SESSION_USER = 'MGILLIS' THEN from_airfield
            WHEN SUBSTR(from_airfield,1,1) = 'E' THEN 'UK'
            ELSE 'XXXX'
      END
ENABLE;
```

This would now allow me, as the DBA, to see all the data that exists in the table in its raw form. However, the user who just inserted a row will still not see their own values. If the requirement is that the user always gets to see their own data regardless of the rest of the Column Mask definition, then the following may be more appropriate:

```
      WHEN SESSION_USER = LAST_UPDATED_BY THEN from_airfield
```

To the second point, you might like to run an EXPLAIN on your query and then a db2exfmt to see what is going on under the covers. The output from db2exfmt will show the **Original Statement** and the **Optimized Statement**. What you will also get in between these two is a section headed **Statement With FGAC Applied.** The user's original query will have changed as shown below



```
Original Statement:
------------------
Select
  LOGBOOK_KEY,
  Date(Brakes_Off) as FLIGHT_DATE,
  REGISTRATION,
  CAPACITY,
  PILOT,
  FROM_AIRFIELD,
  CHECK_PILOT_ID
from
  logbook.logbook
where
  year(Brakes_Off) = '2011'
order by
  Brakes_Off


Statement With FGAC Applied:
---------------------------
SELECT
  Q1.LOGBOOK_KEY AS "LOGBOOK_KEY",
  DATE(Q1.BRAKES_OFF) AS "FLIGHT_DATE",
  Q1.REGISTRATION AS "REGISTRATION",
  Q1.CAPACITY AS "CAPACITY",
  Q1.PILOT AS "PILOT",

CASE
WHEN (SUBSTR(Q1.FROM_AIRFIELD, 1, 1) = 'E')
THEN 'UK '
ELSE 'XXXX' END AS "FROM_AIRFIELD",
     Q1.CHECK_PILOT_ID AS "CHECK_PILOT_ID",
     Q1.BRAKES_OFF
FROM
  LOGBOOK.LOGBOOK AS Q1
WHERE
  (YEAR(Q1.BRAKES_OFF) = 2011)
ORDER BY
  Q1.BRAKES_OFF
```

Note that the column mask CASE statement is now part of your query and will also show up in the Optimized Statement. This shouldn't have any significant effect at this stage (the Timeron value and the access path should be the same), but bear in mind what the optimizer is up to. In the next posting, I will show the relevance of this when Row Permissions are being used.

Part 2

In the previous section we were discussing the Column Mask part of the RCAC feature. To recap, there are 2 parts to RCAC

- Column **masks** : which over-write or obscure returned values with specified alternatives
- Row **permissions** : which only return the rows of data that you are permitted to see

I'm going to try and illustrate some of the concerns that can accompany the use of this feature. I'm going to use the same example as in the previous section: a table called LOGBOOK which contains details of a private pilot's flying log. A user query has been run to look at what was logged in the year 2011. The SQL for this query is shown below

```
Select LOGBOOK_KEY,
       Date(Brakes_Off) as FLIGHT_DATE,
       REGISTRATION,
       CAPACITY,
       PILOT,
       FROM_AIRFIELD,
       CHECK_PILOT_ID
       from LOGBOOK.logbook
       where year(Brakes_Off) = '2011'
       order by Brakes_Off
```

The output from this query would currently be as shown below (Note that the values in the column FROM_AIRFIELD are obfuscated because the Column mask from the previous section is still in operation).

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        232 24.01.2011  G-AYJR       P1       Self                      UK            -
        233 17.04.2011  HB-CFN       P1S      Self                      XXXX          -
        234 30.04.2011  G-AYJR       P1S      Self                      UK            UKFE236606A/A
        235 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        236 21.06.2011  HB-CFN       P1S      Self                      XXXX          -
        237 23.07.2011  G-ETME       P1S      Self                      UK            -
        238 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
        239 28.07.2011  HB-CFN       P1S      Self                      XXXX          -
         14 30.07.2011  G-AVLT       P1S      Self                      XXXX          -
        240 11.10.2011  G-BORK       P1       Self                      UK            -
```

Row Permissions

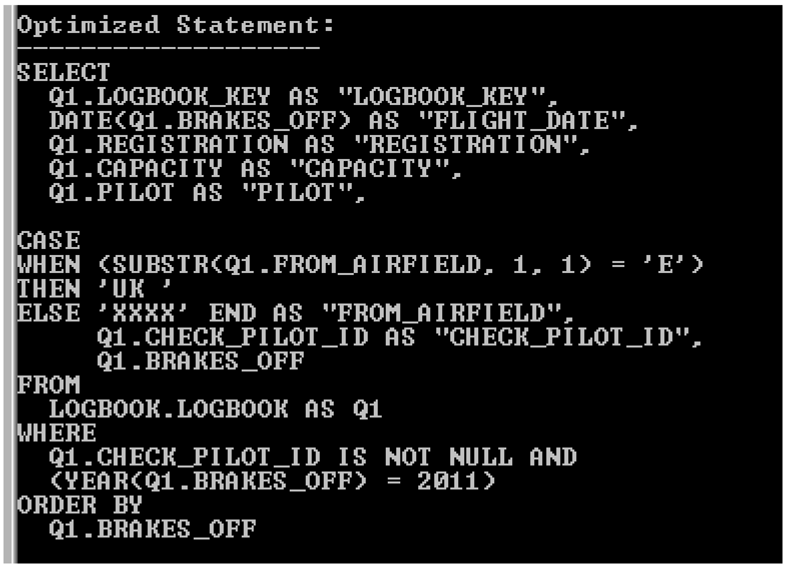
The second task the DBA wants to accomplish is to implement a row permission that will only show the flights with a Check Pilot. This will be created and enabled as follows

```
CREATE or REPLACE PERMISSION CHECK_RIDES ON LogBook FOR ROWS WHERE CHECK_PILOT_ID is not null ENFORCED FOR
ALL ACCESS enable
ALTER TABLE LogBook ACTIVATE ROW ACCESS CONTROL
```

The result set that the user query retrieves will now just be a single row as shown below. Note that the column mask is now operating in conjunction with the row permission, as the Airfield ICAO code is still obfuscated

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        234 30.04.2011  G-AYJR       P1S      Self                      UK            UKFE236606A/A
```

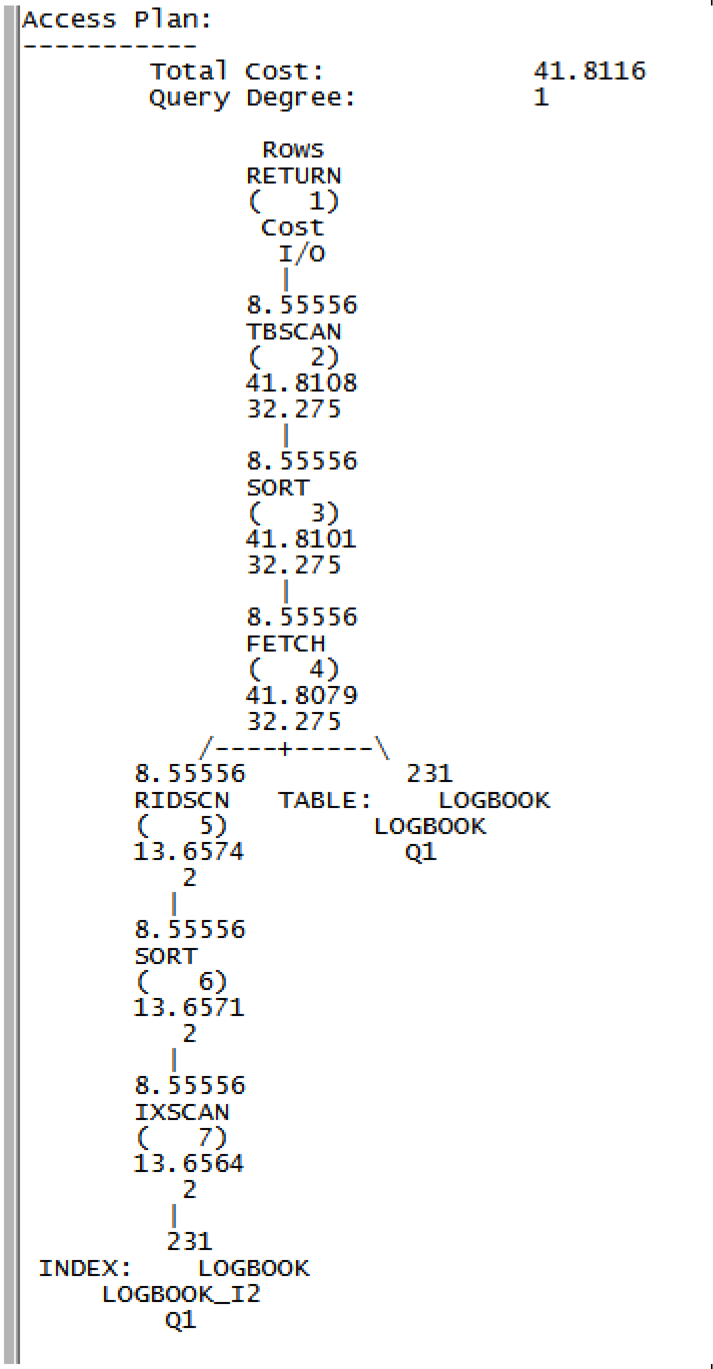
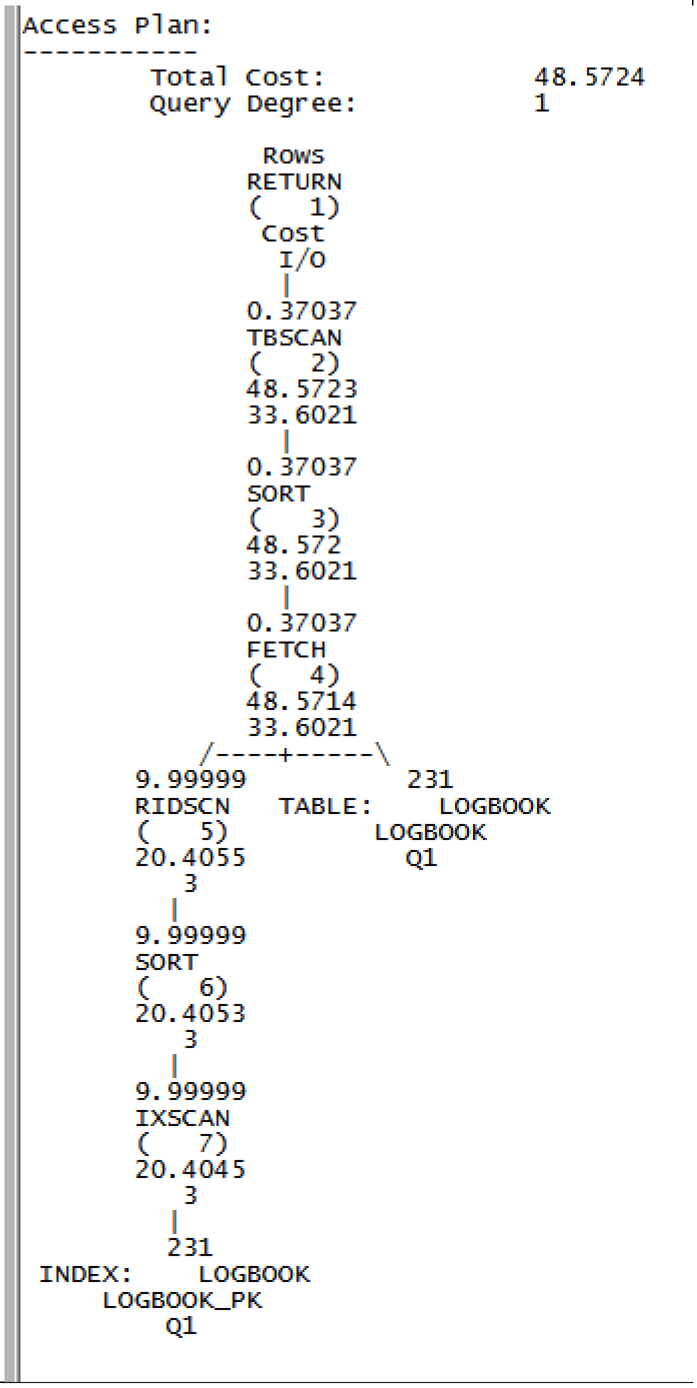
Now take another look at what the optimizer has done under the covers



```
Optimized Statement:
-------------------
SELECT
  Q1.LOGBOOK_KEY AS "LOGBOOK_KEY",
  DATE(Q1.BRAKES_OFF) AS "FLIGHT_DATE",
  Q1.REGISTRATION AS "REGISTRATION",
  Q1.CAPACITY AS "CAPACITY",
  Q1.PILOT AS "PILOT",

CASE
WHEN (SUBSTR(Q1.FROM_AIRFIELD, 1, 1) = 'E')
THEN 'UK '
ELSE 'XXXX' END AS "FROM_AIRFIELD",
     Q1.CHECK_PILOT_ID AS "CHECK_PILOT_ID",
     Q1.BRAKES_OFF
FROM
  LOGBOOK.LOGBOOK AS Q1
WHERE
  Q1.CHECK_PILOT_ID IS NOT NULL AND
  (YEAR(Q1.BRAKES_OFF) = 2011)
ORDER BY
  Q1.BRAKES_OFF
```

The WHERE clause has been modified and a different access path has been invoked.

Original Access Path	Access Path after Row permissions applied
	

```
Access Plan:
-----------
        Total Cost:             41.8116
        Query Degree:           1

              Rows
             RETURN
             (   1)
              Cost
               I/O
               |
             8.55556
             TBSCAN
             (   2)
             41.8108
             32.275
               |
             8.55556
             SORT
             (   3)
             41.8101
             32.275
               |
             8.55556
             FETCH
             (   4)
             41.8079
             32.275
          /----+-----\
      8.55556          231
      RIDSCN   TABLE:    LOGBOOK
      (   5)         LOGBOOK
      13.6574           Q1
         2
        |
      8.55556
      SORT
      (   6)
      13.6571
         2
        |
      8.55556
      IXSCAN
      (   7)
      13.6564
         2
        |
        231
 INDEX:    LOGBOOK
     LOGBOOK_I2
         Q1
```

```
Access Plan:
-----------
        Total Cost:             48.5724
        Query Degree:           1

              Rows
             RETURN
             (   1)
              Cost
               I/O
               |
             0.37037
             TBSCAN
             (   2)
             48.5723
             33.6021
               |
             0.37037
             SORT
             (   3)
             48.572
             33.6021
               |
             0.37037
             FETCH
             (   4)
             48.5714
             33.6021
          /----+-----\
      9.99999          231
      RIDSCN   TABLE:    LOGBOOK
      (   5)         LOGBOOK
      20.4055           Q1
         3
        |
      9.99999
      SORT
      (   6)
      20.4053
         3
        |
      9.99999
      IXSCAN
      (   7)
      20.4045
         3
        |
        231
 INDEX:    LOGBOOK
     LOGBOOK_PK
         Q1
```

The user query execution estimate using the row permission is slightly worse than the original. That may not always be the case but implementing a RCAC could have significant performance considerations and it is worth checking out what the optimizer does to the access plan before releasing it into production. From a user or application point of view, a query with an acceptable execution time may change to unacceptable without the application query having been altered.

Part 3

In the previous two sections we were discussing the Column Mask and Row Permissions as two separate entities, with potentially different impacts to your database. I'm now going to try and demonstrate some of the issues that can occur when you have the two features enabled on the same table.

I'm going to use the same example as in the previous section to illustrate these issues.: a table called LOGBOOK which contains details of a private pilot's flying log. A user query has been run to look at what was logged in the year 2011. The SQL for this query is shown below

```
Select LOGBOOK_KEY,
       Date(Brakes_Off) as FLIGHT_DATE,
       REGISTRATION,
       CAPACITY,
       PILOT,
       FROM_AIRFIELD,
       CHECK_PILOT_ID
       from LOGBOOK.logbook
       where year(Brakes_Off) = '2011'
       order by Brakes_Off
```

The output from this query would currently be as shown below. There are actually 10 rows of data that would qualify without RCAC being in place, but the Row Permissions have only allowed one row to be returned, because only one row has a CHECK_PILOT_ID value. The values in the column FROM_AIRFIELD are altered (the returned row should have the value EGBT) because the Column mask from the first section is still in operation.

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        234 30.04.2011  G-AYJR       P1S      Self                      UK            UKFE236606A/A
```

Masks and Permissions in conjunction

The other little gotcha to bear in mind is how the column mask and the row permissions actually operate. The column mask is being applied to data **after** it has been retrieved and is changing your view of what is there. That's how we are able to insert data that we are not subsequently allowed to see. We could also UPDATE and DELETE data that is protected, or rather, obfuscated by the mask. Not so with the row permission.

Consider this statement

```
Insert into Logbook
(REGISTRATION,CAPACITY,PILOT,FROM_AIRFIELD,TO_AIRFIELD,BRAKES_OFF,BRAKES_ON,Landings,COMMENTS)
Values
('G-AVLT','P1S','Self','EGBT','EGBT','2011-07-30-14.35.00.000000','2011-07-30-15.35.00.000000',6,'TEST
ENTRY')
```

The problem with this is that the data doesn't include a Check Pilot ID, so the result you get is

```
SQL20471N  The INSERT or UPDATE statement failed because a resulting row did not satisfy row permissions.
SQLSTATE=22542
```

That's pretty clear, but what if you try and update or delete a row that you're not supposed to see? I happen to know that there's a row in the table with a Primary Key of 232 (see below for the results of the query ***before*** RCAC was applied):

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        232 24.01.2011  G-AYJR       P1       Self                      EGBT          -
        233 17.04.2011  HB-CFN       P1S      Self                      LFSB          -
        234 30.04.2011  G-AYJR       P1S      Self                      EGBT          UKFE236606A/A
        235 21.06.2011  HB-CFN       P1S      Self                      LFSB          -
        236 21.06.2011  HB-CFN       P1S      Self                      LFGB          -
        237 23.07.2011  G-ETME       P1S      Self                      EGLM          -
        238 28.07.2011  HB-CFN       P1S      Self                      LFSB          -
        239 28.07.2011  HB-CFN       P1S      Self                      LSZQ          -
        240 11.10.2011  G-BORK       P1       Self                      EGBT          -
```

But if I try and delete it, whilst the RCAC Row Permissions are in place, the response is:

```
SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a query is an empty table.
SQLSTATE=02000
```

It is there but we're not allowed to see it. The response is slightly misleading, if logically accurate.

Row permissions are applied ***before*** the query is executed and column masks ***after***. Hence the concern about the access paths when row permissions are applied and hence the slightly misleading responses to UPDATE and DELETE statements.

Removing Masks and Permissions

And thereby hangs the last gotcha I wanted to mention. If the DBA now drops the column mask, the users query results set becomes

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------
        234 30.04.2011  G-AYJR       P1S      Self                      EGBT          UKFE236606A/A
```

Note that the Airfield ICAO code is once again visible, but that the result set only contains a single row because the row permission is still in place. So, now, the DBA drops the row permission and the users result set becomes

```
LOGBOOK_KEY FLIGHT_DATE REGISTRATION CAPACITY PILOT                     FROM_AIRFIELD CHECK_PILOT_ID
----------- ----------- ------------ -------- ------------------------- ------------- ---------------

  0 record(s) selected.
```

This is because there is nothing now to refer to in terms of row permissions, so the default result set will be nothing. The row permissions will be examined to find out ***what the user is allowed to see***; if there are no row permissions, they're not allowed to see anything.

The DBA can remove the RCAC rows and permissions by running the two commands below and all the users data will be visible as it was in the first query we ran.

```
ALTER TABLE LogBook DEACTIVATE ROW ACCESS CONTROL ;
ALTER TABLE LogBook DEACTIVATE COLUMN ACCESS CONTROL;
```

RCAC is a great feature and offers significant security options for the DBA, the business and the application but, like just about everything in the database, it's got something that will come back and bite you if you're not ready for it.

About Triton Consulting
Triton Consulting specialises in Data Management and has been an IBM Premier Business Partner since 1998. Specialising in DB2 for both the mainframe and distributed systems, Triton provides a full range of services from consultancy through to education and 24/7 DB2 support.

For more information visit www.triton.co.uk