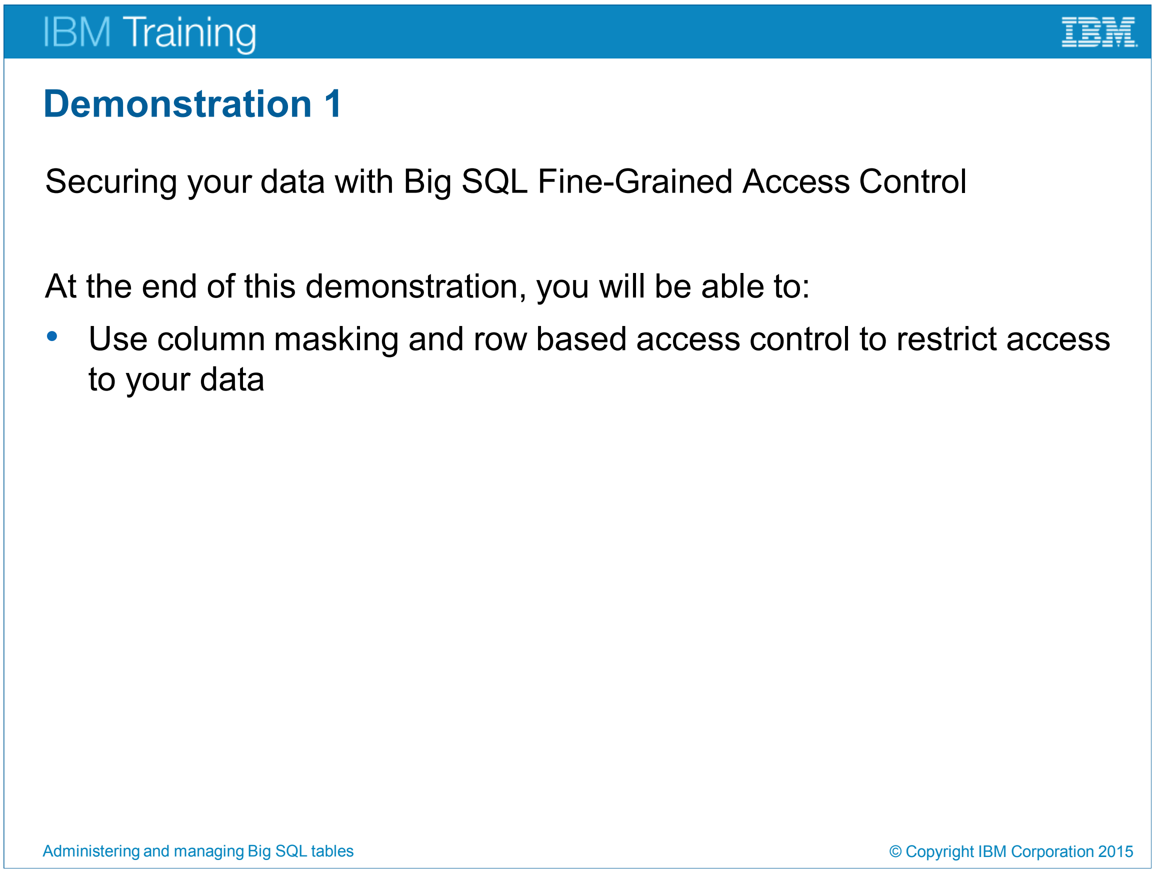


IBM Training

Demonstration 1

Securing your data with Big SQL Fine-Grained Access Control

At the end of this demonstration, you will be able to:

- Use column masking and row based access control to restrict access to your data

Administering and managing Big SQL tables

© Copyright IBM Corporation 2015

Demonstration 1: Securing your data with Big SQL Fine-Grained Access Control

Demonstration 1: Securing your data with Big SQL Fine-Grained Access Control

Purpose:
Big SQL offers administrators additional SQL security control mechanisms for row and column access through the definition of ROLES and GRANT/REVOKE statements. In this demonstration, you will mask information about gross profits for sales from all users who are not a MANAGER. That is, all users with SELECT privileges will be able to query the SLS_SALES_FACT table, but information in the GROSS_PROFIT column will display as 0.0 unless the user was granted the role of a MANAGER. Any user who is a MANAGER will be able to see the underlying data values for GROSS_PROFIT.

To complete this demonstration, you must have access to multiple user accounts. Examples in this demonstration are based on the following user IDs (in addition to biadmin):

bigsql, which has SECADM authority for your database environment.

user1 and *user2*, which have USER privileges for BigInsights. The password for these two are simply: password

In addition, prior to starting this demonstration, you must have created the SLS_SALES_FACT and the MRK_PROMOTION_FACT table and populated it with data, as described in an earlier demonstration. Section 1 will show you how to load the table if you do not have it created and loaded.

Estimated time: 30 minutes

User/Password:	**biadmin/biadmin**
	root/dalvm3
	user1/password
	user2/password
	bigsql/ibm2blue
Services Password:	**ibm2blue**

Task 1. Ensure that BigInsights is running and the Big SQL tables have been created and populated.

You may skip this task if your environment is still up and running. Refer to past demonstrations if you need to do any of these.

1. Ensure BigInsights is running via Ambari.
2. Ensure you can access the BigInsights Home page
3. Ensure you can access the Big SQL page.
4. The tables SLS_SALES_FACT and MRK_PROMOTION_FACT should have been created and populated with data from a previous exercise. If not, use the script provided under /home/biadmin/labfiles/bigsql/ and complete this now.

Task 2. Using column masking to secure data.

The SQL statements are provided in a script located under **/home/biadmin/labfiles/bigsql/Big_SQL_FGAC/**

You will create two new users, **user1** and **user2** both with the **password** as the password.

1. Switch to the **root** user with **dalvm3**

```
su -
```

2. Create **user1** and **user2**

```
adduser user1
adduser user2
```

3. Set the password as **password**

```
passwd user1
passwd user2
```

Create two roles. These commands must be executed by the bigsql user.

4. Using either JSqsh or the Big SQL console, create these two roles:

```
CREATE ROLE manager;
CREATE ROLE staff;
```

5. Grant SELECT (read) access to the table to users.

```
GRANT SELECT ON sls_sales_fact TO USER user1;
GRANT SELECT ON sls_sales_fact TO USER user2;
```

6. Issue GRANT statements that assign appropriate roles to desired users.

```
GRANT ROLE MANAGER TO USER user1;
GRANT ROLE STAFF TO USER user2;
```

7. Create a column access control rule. Specifically, create a mask called PROFIT_MASK for the GROSS_PROFIT column of the MYBIGSQL.SLS_SALES_FACT table that will display a value of 0.0 to any user who queries this column that is not a MANAGER.

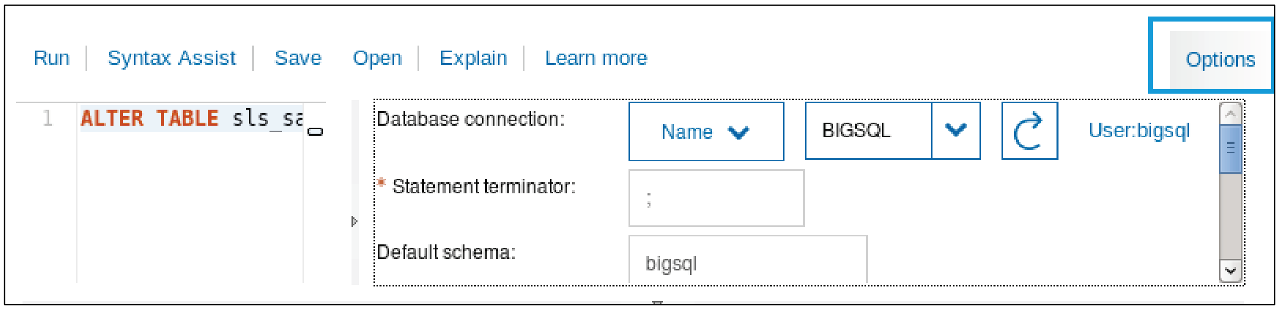
```
CREATE MASK PROFIT_MASK ON
sls_sales_fact
FOR COLUMN gross_profit
RETURN
CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'MANAGER') =
1
THEN gross_profit
ELSE 0.0
END
ENABLE;
```

8. Issue the following ALTER TABLE statement to activate the column based access control restriction.

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

Now you are ready to test the results of your work by querying the SLS_SALES_FACT table using different user accounts. You will do this using the Big SQL page.

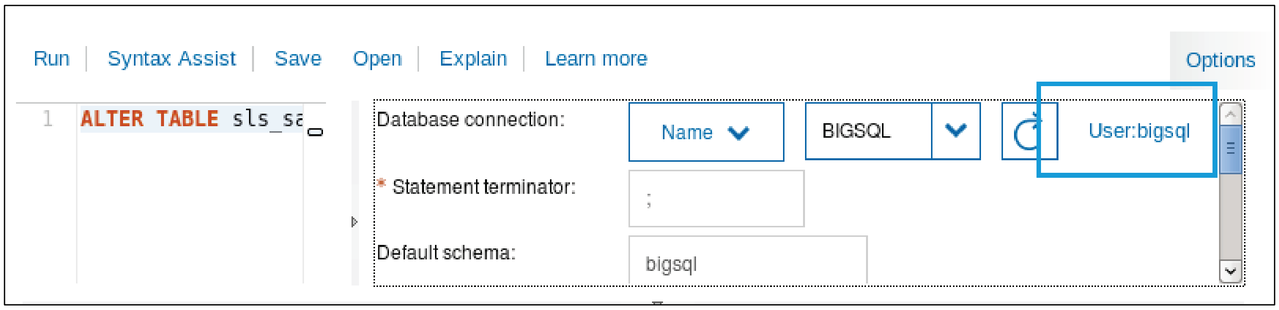
9. Click on the **Options** tab.



```
ALTER TABLE sls_sa
```

Database connection: Name BIGSQL User:bigsql
* Statement terminator: ;
Default schema: bigsql

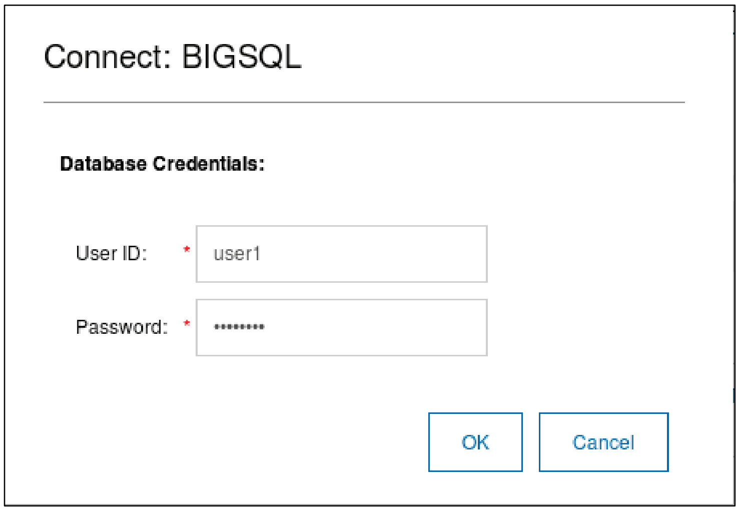
10. Click on the **user:bigsql** link to change to another user.



```
ALTER TABLE sls_sa
```

Database connection: BIGSQL
* Statement terminator: ;
Default schema: bigsql

11. Change to the **user1** account. The password is **password**.



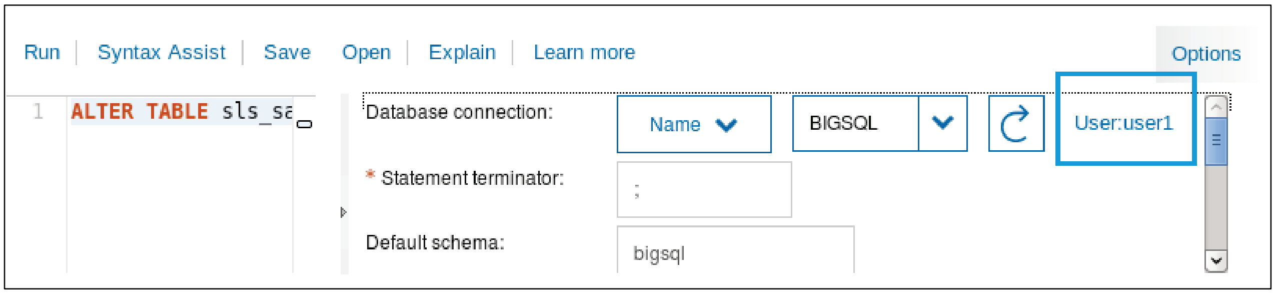
Connect: BIGSQL

Database Credentials:

User ID: * user1

Password: * ••••••••

12. Once you have logged in as **user1**, the link should reflect that.



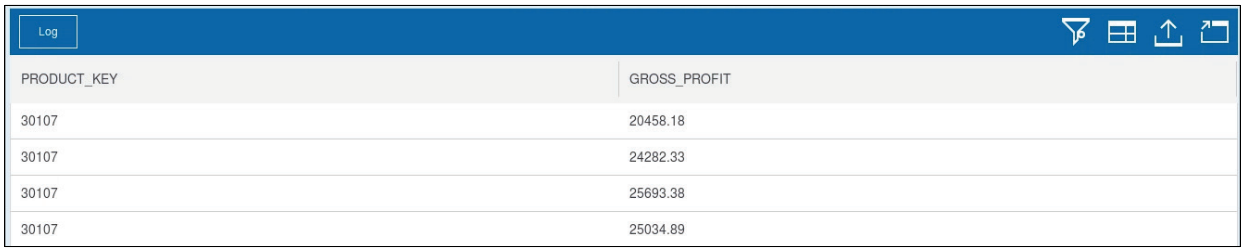
```
ALTER TABLE sls_sa
```

Database connection: BIGSQL
* Statement terminator: ;
Default schema: bigsql

13. Collapse the **Options** pane.
14. Run the following query as user1.

```
select product_key, gross_profit
from sls_sales_fact
where quantity > 5000 fetch first 5 rows only;
```

15. Inspect the results, and note that various data values appear in the GROSS_PROFIT column. This is what you should expect, because USER1 is a MANAGER, and your column mask rule allows MANAGERs to see the data values for this column.

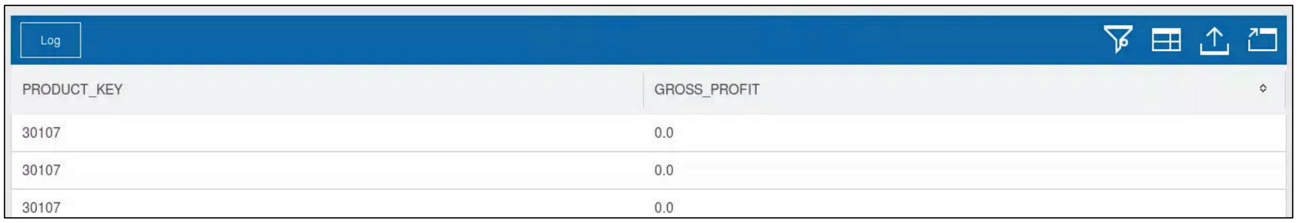


PRODUCT_KEY	GROSS_PROFIT
30107	20458.18
30107	24282.33
30107	25693.38
30107	25034.89

16. Using the same steps as before, switch to **user2/password.**
17. Run the same query as before.

```
select product_key, gross_profit
from sls_sales_fact
where quantity > 5000 fetch first 5 rows only;
```

18. Inspect the results. Note that the GROSS_PROFIT values are masked (appearing as 0.0). Again, this is what you should expect, because USER2 has a STAFF role. Any users who are not MANAGERs are not allowed to see values for this column.



PRODUCT_KEY	GROSS_PROFIT
30107	0.0
30107	0.0
30107	0.0

19. Switch to the bigsql account and deactivate the column access restriction.

```
ALTER TABLE sls_sales_fact DEACTIVATE COLUMN ACCESS
CONTROL;
```

Task 3. Using row-based access control.

The effort required to implement row-based access control rules is similar. You will explore a row-based scenario now using the MRK_PROMOTION_FACT table you created and populated in an earlier lab (or in this lab). After implementing this example, you'll see that SELECT statements issued by user1 will only return rows related to a specific retailer key. Restrict consultants (CONSULT role users) to accessing only rows. For retailer key 7166 commands must be executed from bigsql user ID. A valid ID must exist for user1. In this demonstration, user1 is a consultant.

1. Create a new role named CONSULT.

```
CREATE ROLE CONSULT;
```

2. Grant SELECT (read) access to the table to user1 and user2.

```
GRANT SELECT ON mrk_promotion_fact TO USER user1;
GRANT SELECT ON mrk_promotion_fact TO USER user2;
```

3. Issue GRANT statements that assign appropriate roles to desired users. In this case, assign user1 CONSULT role. Do not assign any role to user2.

```
GRANT ROLE CONSULT TO USER user1;
```

4. Create a row access control rule. Specifically, restrict read operations on mybigsql.mrk_promotion_fact to users with the CONSULT role. Furthermore, allow such users to only see rows in which RETAILER_KEY column values are 7166.

```
CREATE PERMISSION RETAILER_7166
ON mrk_promotion_fact
FOR ROWS
WHERE(VERIFY_ROLE_FOR_USER(SESSION_USER,'CONSULT') = 1
AND
retailer_key = 7166)
ENFORCED FOR ALL ACCESS
ENABLE;
```

5. Issue the following ALTER TABLE statement to activate the row based access control restriction.

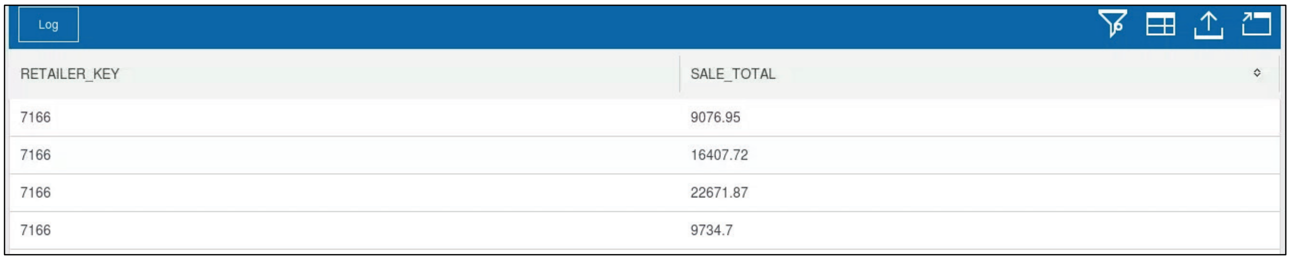
```
ALTER TABLE mrk_promotion_fact ACTIVATE ROW ACCESS
CONTROL;
```

Now you are ready to test the results of your work by querying the table.

6. Switch to the **user1** user with the password = **password**.
7. Run the query as user1:

```
select retailer_key, sale_total
from mrk_promotion_fact
where rtl_country_key = 90010
fetch first 100 rows only;
```

8. Inspect the results, and note that only rows for retailer 7166 appear. This is what you should expect, because USER1 is associated with the CONSULT row.



RETAILER_KEY	SALE_TOTAL
7166	9076.95
7166	16407.72
7166	22671.87
7166	9734.7

9. Switch to **user2** with the password = **password**.
10. Run the query again.

```
select retailer_key, sale_total
from mrk_promotion_fact
where rtl_country_key = 90010
fetch first 100 rows only;
```

11. Inspect the results. Note that the query runs successfully but returns no rows. Why? The row access control mechanism you implemented specified that only users of the CONSULT row are permitted to see data from the data and that data would be restricted to rows related to a specific RETAILER_KEY value.
12. Switch back to the **bigsql** user and deactivate the access control:

```
ALTER TABLE mrk_promotion_fact DEACTIVATE ROW ACCESS
CONTROL;
```

Results:
In this demonstration, you masked information about gross profits for sales from all users who are not a MANAGER. You accessed multiple user accounts, including *bigsql*, which has SECADM authority for your database environment.