**Table of Contents**

[**ABSTRACT** 3](#_Toc45901463)

[**Functionality** 3](#_Toc45901464)

[Procedures 5](#_Toc45901465)

[Tables 6](#_Toc45901466)

[ER Diagram 6](#_Toc45901467)

[Codes 7](#_Toc45901468)

[**Auditing Model** 9](#_Toc45901469)

[**Virtual Private Database (VPD)** 9](#_Toc45901470)

[Implementation & Demonstration 9](#_Toc45901471)

[Virtual Private Database (VPD) Implementation 10](#_Toc45901472)

[Table structure modifications 11](#_Toc45901473)

[Virtual Private Database – DEMONSTRATION (Policy Enabled) 19](#_Toc45901474)

[Virtual Private Database – DEMONSTRATION (Policy DISABLED) 24](#_Toc45901475)

[**Implementation of the Auditing Solution** 26](#_Toc45901476)

[Actions Audited by Default 26](#_Toc45901477)

[Steps taken to enable Unified Auditing- 27](#_Toc45901478)

Audit Failed Logins………………………………………………………………………………..29

Audit Privilages……………………………………………………………………………………31

Audit Book Search………………………………………………………………………………..35

Audit Kiosk ……………………………………………………………………………….............39

Audit Transactions……………………………………………………………………………..…41

Audit User Deletion…………………………………………………………………….…………44

**ABSTRACT**

This project designs a database system for KSU Book Kiosks. It contains a process, consisting of stored procedures that checks usernames against a list of registered users. If the users are authorized, they granted access to search book collections either by name or ISBN number. From that point, the user can request the desired book or return a book when it is due. Administrators are authorized to print weekly reports for each kiosk.

**Functionality**

1. Upon accessing the KSU Library Database, the user’s username is checked against a list of registered users within the KSU Library database. If the user is registered, they are granted access to the library databases. Otherwise, they are directed to register.

USER CATEGORIES:

* *Unregistered* – user is denied access into the KSU Library Database.
* *Registered* – user is granted access into the KSU Library Database
* *Administrator* – user has admin privileges to update and delete all tables within the KSU Library Database.

1. If a user logs in with an incorrect passcode 3 times, the username is locked, and the need for DBA to unlock is required.
2. Upon logging into the system, various actions can be made by the registered user:
3. **General Search for a Book (by ISBN)**

* The user can search for books based on the ISBN as input parameter.
* The book will be available for checkout if the Book Quantity is > 0.
* The book will be unavailable for checkout if either the Book Quantity = 0 or does not exist in the database.

1. **General Search for a Book (by Book Title)**

* The user can search for books based on the Book Title.
* The book will be available for checkout if the Book Quantity is > 0.
* The book will be unavailable for checkout if either the Book Quantity = 0 or does not exist in the database.

1. **Checking Out a Book**

* A user can request to checkout a book if available.
* If user logs in and receives notification of an existing late fee, that means they exist on the EXCEPTIONS table with a balance. They will get an error informing them to pay balance before proceeding. At this point, they can only update the amount on the exceptions table by executing PAY\_FEE (demonstrating a balance payment). Once the fee balance is updated to $0.00, they can continue with checkout.
* If user does not have a fee, they are granted access.
* User is notified if book is not available for checkout.
* After a successful checkout:
  1. The user receives a receipt of the transaction specifying due date and notice of the $5.00 flat rate late return fee.
  2. A record of the user’s transaction is inserted into the TRANSACTIONS table
     + Timestamp recorded
     + Due Date is set to System Date + 30 Days
     + Status is set to CHECKED OUT
  3. The book’s quantity is updated in the KIOSK table.

1. **Returning a Book**

* A user can request to return a book if the book is in Checkout Status.
* An exception is received if the user inputs an ISBN that is not checked out.
* If the user is returning the book before the due date:
  1. The user’s transaction record is updated in the TRANSACTIONS table
     1. Timestamp recorded
     2. Return Date is updated to System Date
     3. Status is set to RETURNED
  2. The book’s quantity is updated in the KIOSK table
* If the user is returning the book after the due date:
  1. The user receives an alert that the book is late along with the flat rate late fee of $5.00.
  2. A row is inserted into the EXCEPTIONS table recording the late fee and revoking checkout privileges.
  3. The user’s transaction record is updated in the TRANSACTIONS table
     1. Timestamp is recorded
     2. Return Date is updated to System Date
     3. Status is set to RETURNED
  4. The book’s quantity is updated in the KIOSK table.

1. **Updating Kiosk/Book Information**

* An administrator is the only user with access to update either KIOSK or BOOK table to change any information outside of the normal checkout transaction. Any other user attempting to perform this action will receive an error.
* Before the administrator updates the kiosk/book info, the system checks if the ISBN exists in the system.
* If the ISBN exists:
  1. The book quantity in the KIOSK table is increased by the user’s input number.
  2. A receipt of the transaction is provided to the user summarizing the update.
* If the ISBN does not exist:
  1. The administrator receives notification that the book does not exist in the system.

1. **Deleting a Technician**

* An administrator is the only user with access to delete a technician from the system.
* Any other user attempting to perform this action will receive an error.

1. **Unauthorized Permissions**

* All user permissions will be tested:

Registered Users, Unregistered users and administrators.

1. **Weekly Report Generation**

* The administrator will run reports against each kiosk to get detailed activity/transactions for that week.

### Procedures

BOOKSEARCHISBN

- Users search a book by ISBN

Input Parameter

* ISBN

BOOKSEARCHTITLE

- Users search a book by title

Input Parameter

* TITLE

CHECK\_OUT\_BOOK

- Users checkout a book

Input Parameters

* USERID
* ISBN

PAY\_FEE

* Users with late fee runs this to pay fee in full

Input Parameters

* USERID

CHECK\_IN\_BOOK

* Users return a book

Input Parameters

* USERID
* ISBN

UPDATE\_BOOK\_KIOSKS

* Admin updates the Kiosk/Book Information

Input Parameters

* ISBN
* Quantity of new books to add or remove

DROP\_USER

* Deletes a Technician

Input Parameters

* USERNAME

### Tables

USERS – stores all users who are registered within the bookstore database.

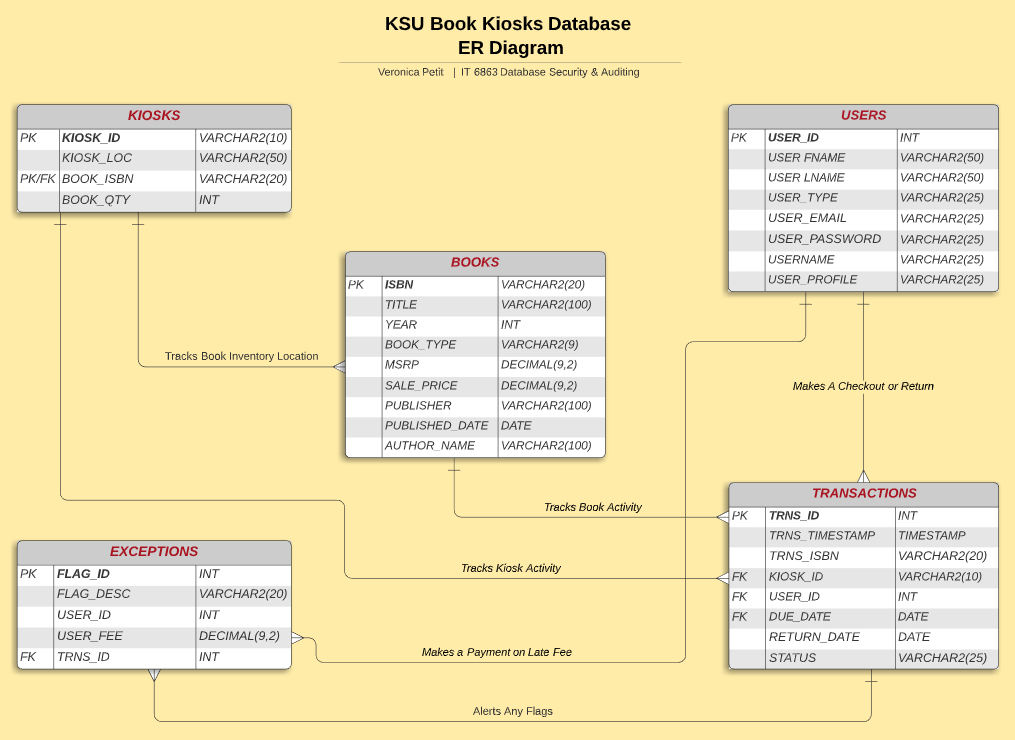
KIOSKS – stores all bookstore kiosk locations along with the inventory of each book.

BOOKS – stores all books available to the bookstore kiosks.

TRANSACTIONS – stores and tracks all book checkouts and returns.

EXCEPTIONS – audit table to keep track of any user who incurs a late return fee.

### ER Diagram



### Codes

**-- -------------------------------------------------**

**-- CREATE TABLES**

**-- -------------------------------------------------**

**CREATE TABLE USERS (**

USER\_ID INT GENERATED ALWAYS AS IDENTITY (START WITH 100001 INCREMENT BY 1) NOT NULL,

USER\_FIRSTNAME VARCHAR2(15),

USER\_LASTNAME VARCHAR2(15),

USER\_EMAIL VARCHAR2(30),

USER\_PHONE VARCHAR2(15),

USER\_PASSWORD VARCHAR2(15),

USER\_TYPE VARCHAR2(10),

USER\_PROFILE VARCHAR2(10),

USERNAME VARCHAR2(15)

);

**CREATE TABLE BOOKS (**

ISBN VARCHAR2(20) NOT NULL,

TITLE VARCHAR2(100),

YEAR INT,

BOOK\_TYPE VARCHAR2(9),

MSRP DECIMAL(9,2),

PUBLISHER VARCHAR2(100),

PUBLISHED\_DATE DATE,

AUTHOR\_NAME VARCHAR2(100)

);

**CREATE TABLE KIOSKS (**

KIOSK\_ID VARCHAR2(10) NOT NULL,

KIOSK\_LOC VARCHAR2(50),

BOOK\_ISBN VARCHAR2(20) NOT NULL,

BOOK\_QTY INT,

CTRL\_USER VARCHAR2(25)

);

**CREATE TABLE TRANSACTIONS (**

TRNS\_ID INT GENERATED ALWAYS AS IDENTITY (START WITH 300001 INCREMENT BY 1) NOT NULL,

TRNS\_TIMESTAMP TIMESTAMP,

TRNS\_ISBN VARCHAR2(20),

KIOSK\_ID VARCHAR2(10),

USER\_ID INT,

DUE\_DATE DATE,

RETURN\_DATE DATE,

STATUS VARCHAR2(25),

CTRL\_USER VARCHAR2(25)

);

**CREATE TABLE EXCEPTIONS (**

FLAG\_ID INT GENERATED ALWAYS AS IDENTITY (START WITH 500001 INCREMENT BY 1) NOT NULL,

FLAG\_DESC VARCHAR2(20),

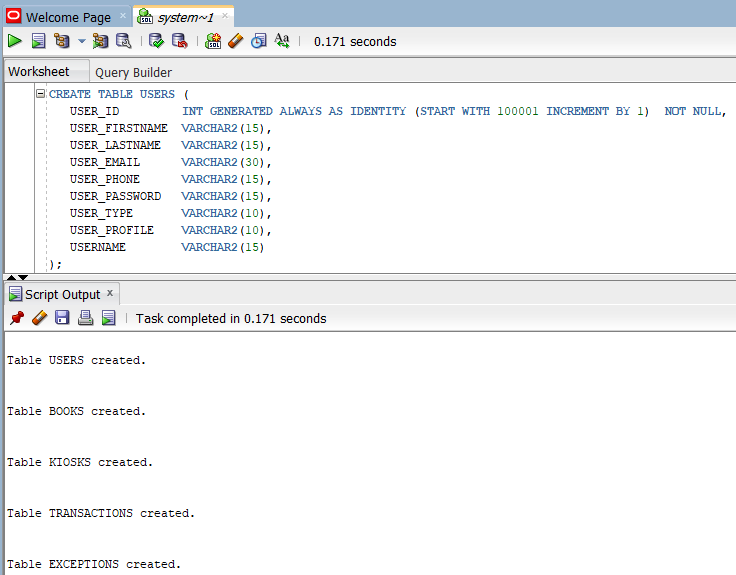
TRNS\_ID INT NOT NULL,

USER\_ID INT NOT NULL,

USER\_FEE DECIMAL(9,2),

CTRL\_USER VARCHAR2(25)

);



**Auditing Model**

Auditing is the process of monitoring and recording the user activities and changes made in the data through user activities which include the execution of statements and stored procedures. Our group project deals with the library database system at KSU Kiosk with information about the books as well as users. There are three types of users can work in the database: administrators, technician and student users. In our project, we will audit all the actions taken by the database users and nondatabase users in order to ensure that our database security policies are correctly implemented, and the data are protected from unauthorized uses. We also want to enable accountability for actions. While we may not be able to audit every table, we want to audit those columns which contain the most vulnerable data.

We are using the unified auditing model to use in this group project. All the audit records will be stored in the unified audit trail in a uniform format. We want to use the unified auditing model because it can capture audit records from a various sources such as the audit records including Sys audit records, fine grained audit records, oracle database real application serenity audit records and oracle recovery manage records. This model has the ability to create an opportunity for us to analyze data from multiple sources, giving room to see ways to improve and initiate a better system. The management and security of the audit trail is also improved by having it in single audit trail. This model does not depend on the initialization parameters. Once enabled, Unified auditing frees you from setting different initialization parameters. Auditing is always turned-on.

Additionally, the audit records, including records from the SYS audit trail, for all the audited components of your Oracle Database installation are placed in one location and in one format, rather than your having to look in different places to find audit trails in varying formats. This consolidated view enables auditors to co-relate audit information from different components. All data is stored in secured files, not even the SYS user can tamper with them. This simplistic setup creates a consolidated view that will allow us to audit more efficiently. As for example, if there is a system error while executing the select statement, the unified audit model will write a unique code to the audit trail.

The implementation is very convenient. Oracle 12c and beyond provides the dbms\_audit\_mgmt package for managing audit information. Oracle now has "create audit policy" syntax to easily create audit roles.

**Virtual Private Database (VPD)**

### Implementation & Demonstration

**Instructions to recreate the demonstration:**

1.) Login into Oracle SQL Developer using the normal SYSTEM login credentials:

login = SYSTEM

password = oracle18cdb

2.) Run the command to confirm you are on the root container

alter sessions set container = CDB$ROOT;

2.) Open the .sql file (script) and execute it.

3.) The script will load the entire KSU Book Kiosk Database (tables, data, users, procedures, etc.)

4.) Once script completes, you are now ready to use the KSU Book Kiosk Database!

Below are the bookstore users created for demonstration.

All users are common users, with all using the same password = oracle18cdb.

Because these are all common users so it’s important you are on the CDB$ROOT container.

**\*\*\*\* Do Not change/alter the container to samplescdb \*\*\*\***

**Student (Unregistered User)**

C##SBASS

**Student (Registered Users)**

C##MTHOMAS

C##LMILLER

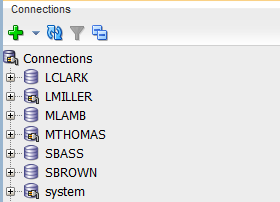
C##SBROWN

**Employee (Administrator)**

C##LCLARK

**Employee (Non-Administrator)**

C##MLAMB



### Virtual Private Database (VPD) Implementation

The KSU Book Kiosk Database incorporates a Virtual Private Database concept.

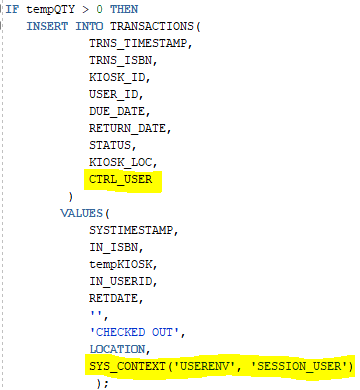
Our VPD was implemented using system context and row filters (RLS).

### Table structure modifications

The impacted tables were updates to support VPD/RLS concepts by appending a *control* column that would be used in row filtering. Each row was updated to include the user who has access to view it.

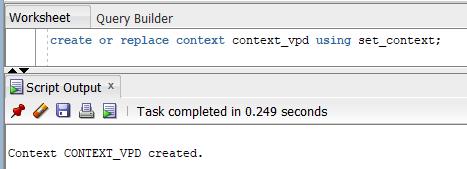
Below are the detailed steps used to implement our VPD concept:

1. **Appended columns to the tables enabled to the Virtual Private Database policy.**
   1. USERS - *username* is the control column
      1. managers can view all rows
      2. users can only view rows containing their user id
   2. TRANSACTIONS - *ctrl\_user* is the control column
      1. manager can view all rows
      2. user can only view rows containing their user id
2. **Updated the CHECK\_OUT\_BOOK Procedure to Insert the Username into the new CTRL\_USER Column once checkout is complete.**



1. **Created context using set\_context**

Create or Replace context context\_vpd using set\_context;



1. **Created procedure to call context**

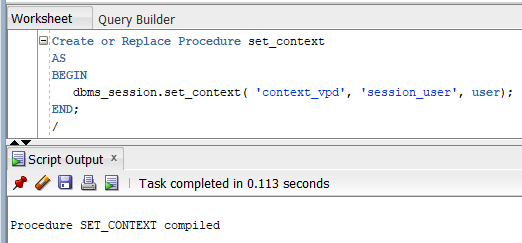
Create or Replace Procedure set\_context

AS

BEGIN

dbms\_session.set\_context('context\_vpd', 'session\_user', user);

END;



1. **Created trigger to apply set\_context upon user logon**

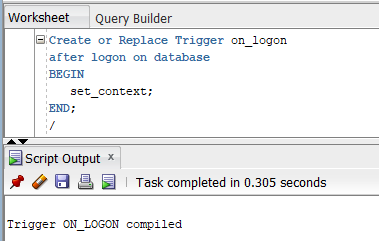
Create or Replace Trigger on\_logon

after logon on database

BEGIN

set\_context;

END;



1. **Created the function that will modify the query for VPD enabling on TRANSACTIONS table**

Create or Replace FUNCTION security\_function (obj\_schema varchar2, obj\_name varchar2)

RETURN VARCHAR2

AUTHID DEFINER

AS

returned\_string VARCHAR2(100);

usertype VARCHAR2(10);

BEGIN

SELECT USER\_TYPE

INTO USERTYPE

FROM SYSTEM.USERS

WHERE USERNAME = UPPER(SYS\_CONTEXT('USERENV', 'SESSION\_USER'));

-- IF USER IS AN EMPLOYEE ON THE KSU KIOSK DATABASE, THEY CAN VIEW ALL ROWS ON THE TRANSACTIONS TABLE

IF usertype = 'EMPLOYEE’ THEN

returned\_string := 'ctrl\_user '||'IS NOT NULL';

return returned\_string;

ELSE

-- IF USER IS NOT AN EMPLOYEE THEN USER CAN ONLY VIEW THE ROWS FOR THEIR USERNAME

returned\_string := 'ctrl\_user= ''' || UPPER(SYS\_CONTEXT('USERENV', 'SESSION\_USER')) || '''';

return returned\_string;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- IF USER IS NOT AN EMPLOYEE THEN CHECK IF SYSTEM (CAN VIEW ALL ROWS ON DATABASE)

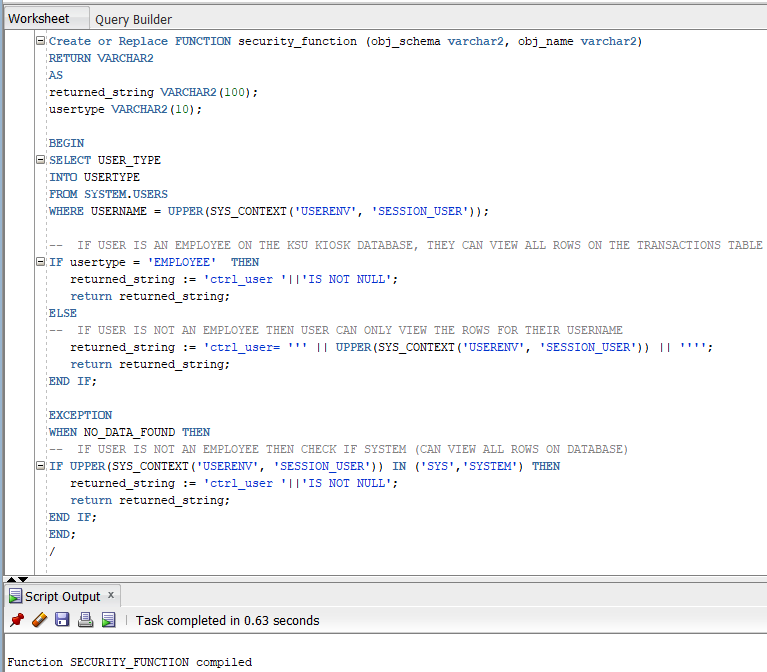
IF UPPER(SYS\_CONTEXT('USERENV', 'SESSION\_USER')) IN ('SYS','SYSTEM') THEN

returned\_string := 'ctrl\_user '||'IS NOT NULL';

return returned\_string;

END IF;

END;



1. **Create the function that will modify the query for VPD enabling on USERS table**

Create or Replace FUNCTION security\_function2 (obj\_schema varchar2, obj\_name varchar2)

RETURN VARCHAR2

AS

returned\_string VARCHAR2(100);

NAME VARCHAR2(50) DEFAULT 'USERNAME IS NOT NULL';

BEGIN

IF UPPER(SYS\_CONTEXT('USERENV', 'SESSION\_USER')) IN('SYSTEM','SYS','C##LCLARK') THEN

-- IF USER IS AN EMPLOYEE ON THE KSU KIOSK DATABASE, THEY CAN VIEW ALL ROWS ON THE TRANSACTIONS TABLE

returned\_string := NAME;

return returned\_string;

ELSE

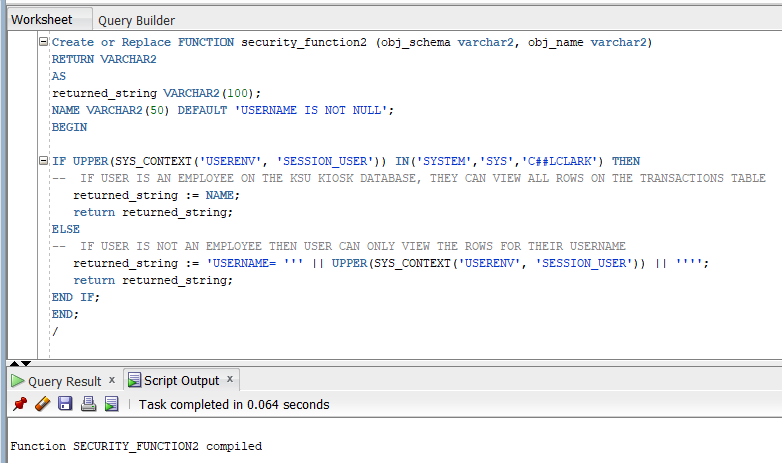
-- IF USER IS NOT AN EMPLOYEE THEN USER CAN ONLY VIEW THE ROWS FOR THEIR USERNAME

returned\_string := 'USERNAME= ''' || UPPER(SYS\_CONTEXT('USERENV', 'SESSION\_USER')) || '''';

return returned\_string;

END IF;

END;



1. **Enabled the policies**

**TRANSACTIONS TABLE**

BEGIN

dbms\_rls.add\_policy(object\_schema => 'system',

object\_name => 'users',

policy\_name => 'users\_policy',

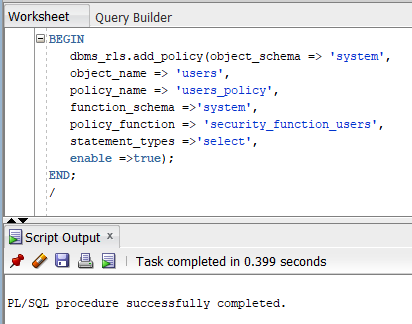
function\_schema =>'system',

policy\_function => 'security\_function2',

statement\_types =>'select',

enable =>true);

END;



**USERS TABLE**

BEGIN

dbms\_rls.add\_policy(object\_schema => 'system',

object\_name => 'transactions',

policy\_name => 'transactions\_policy',

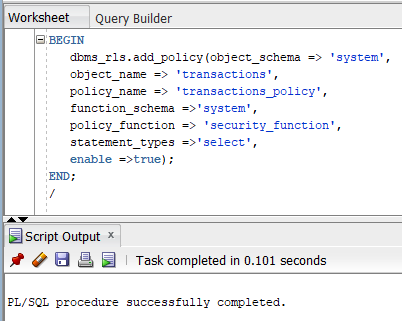
function\_schema =>'system',

policy\_function => 'security\_function',

statement\_types =>'select',

enable =>true);

END;



1. **Granted select and execute permissions on the tables to the users/roles.**

-- -----------------------------------

-- CREATE AND GRANT ROLE FOR EMPLOYEES (Admins)

-- ------------------------------------

CREATE ROLE C##EMPLOYEES;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.USERS TO C##EMPLOYEES;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.BOOKS TO C##EMPLOYEES;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.KIOSKS TO C##EMPLOYEES;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.EXCEPTIONS TO C##EMPLOYEES;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.TRANSACTIONS TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHISBN TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHTITLE TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.CHECK\_IN\_BOOK TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.CHECK\_OUT\_BOOK TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.UPDATE\_BOOK\_KIOSKS TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.PAY\_FEE TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.SEARCH\_TRANSACTIONS TO C##EMPLOYEES;

GRANT EXECUTE ON SYSTEM.SEARCH\_USERS TO C##EMPLOYEES;

GRANT C##EMPLOYEES TO C##LCLARK;

-- -----------------------------------

-- CREATE AND GRANT ROLE FOR EMPLOYEES (Non-Admins)

-- ------------------------------------

CREATE ROLE C##EMPLOYEES2;

GRANT SELECT, INSERT, UPDATE ON SYSTEM.USERS TO C##EMPLOYEES2;

GRANT SELECT, INSERT, UPDATE ON SYSTEM.BOOKS TO C##EMPLOYEES2;

GRANT SELECT, INSERT, UPDATE ON SYSTEM.KIOSKS TO C##EMPLOYEES2;

GRANT SELECT, INSERT, UPDATE ON SYSTEM.EXCEPTIONS TO C##EMPLOYEES2;

GRANT SELECT, INSERT, UPDATE ON SYSTEM.TRANSACTIONS TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHISBN TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHTITLE TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.CHECK\_IN\_BOOK TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.CHECK\_OUT\_BOOK TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.UPDATE\_BOOK\_KIOSKS TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.PAY\_FEE TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.SEARCH\_TRANSACTIONS TO C##EMPLOYEES2;

GRANT EXECUTE ON SYSTEM.SEARCH\_USERS TO C##EMPLOYEES2;

GRANT C##EMPLOYEES2 TO C##MLAMB;

-- -----------------------------------

-- CREATE AND GRANT ROLE FOR STUDENTS

-- ------------------------------------

CREATE ROLE C##STUDENTS;

GRANT SELECT, INSERT ON SYSTEM.USERS TO C##STUDENTS;

GRANT SELECT, INSERT ON SYSTEM.BOOKS TO C##STUDENTS;

GRANT SELECT, INSERT ON SYSTEM.KIOSKS TO C##STUDENTS;

GRANT SELECT, INSERT ON SYSTEM.TRANSACTIONS TO C##STUDENTS;

GRANT SELECT, INSERT ON SYSTEM.EXCEPTIONS TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHISBN TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.BOOKSEARCHTITLE TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.CHECK\_IN\_BOOK TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.CHECK\_OUT\_BOOK TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.PAY\_FEE TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.SEARCH\_TRANSACTIONS TO C##STUDENTS;

GRANT EXECUTE ON SYSTEM.SEARCH\_USERS TO C##STUDENTS;

GRANT C##STUDENTS TO C##MTHOMAS;

GRANT C##STUDENTS TO C##SBASS;

GRANT C##STUDENTS TO C##LMILLER;

GRANT C##STUDENTS TO C##SBROWN;

1. **Created Procedures to test VPD policies - SELECT \* From Tables**

* **(Will confirm user can only view based on RLS Filtering)**

**TRANSACTIONS TABLE**

Create or Replace PROCEDURE Search\_Transactions

AUTHID DEFINER

IS

RC SYS\_REFCURSOR;

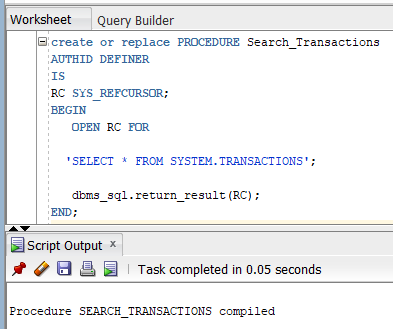
BEGIN

OPEN RC FOR

'SELECT \* FROM SYSTEM.TRANSACTIONS';

dbms\_sql.return\_result(RC);

END;



**USERS TABLE**

create or replace PROCEDURE Search\_Users

AUTHID DEFINER

IS

RC SYS\_REFCURSOR;

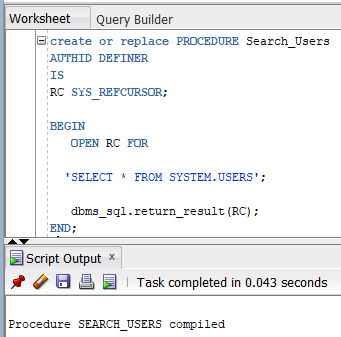
BEGIN

OPEN RC FOR

'SELECT \* FROM SYSTEM.USERS';

dbms\_sql.return\_result(RC);

END;



### Virtual Private Database – DEMONSTRATION (Policy Enabled)

**TRANSACTIONS TABLE:**

**Test Case #1:**

Scenario = Manager wants to view all rows on TRANSACTION table

User Type = (Employee Administrator)

Username = C##LCLARK

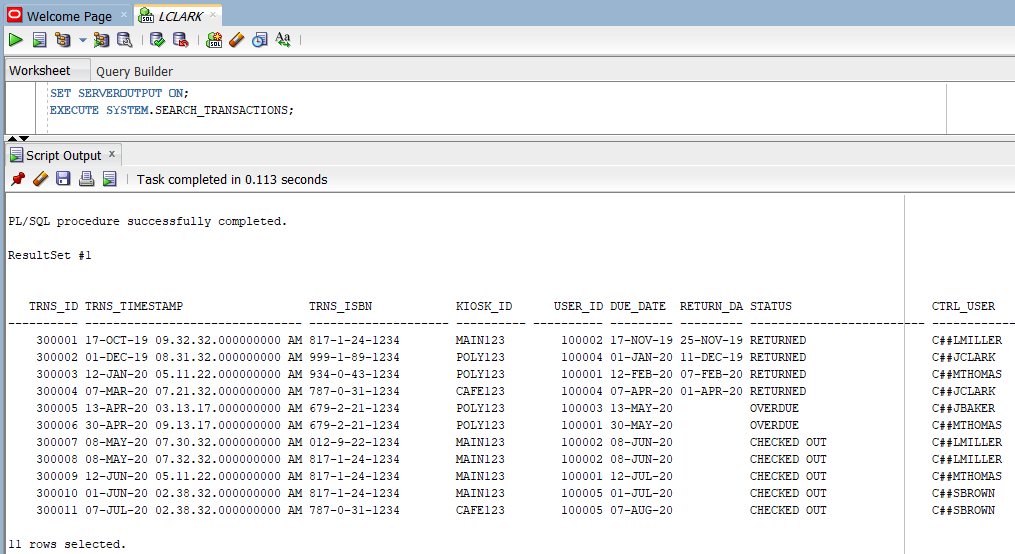
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_TRANSACTIONS;

Expected Result = Administrators have access to view ALL rows on the table.

Actual Result = User can view all rows.



**Test Case #2:**

Scenario = User wants to view data on the TRANSACTIONS table

User Type = (Registered Student)

Username = C##MTHOMAS

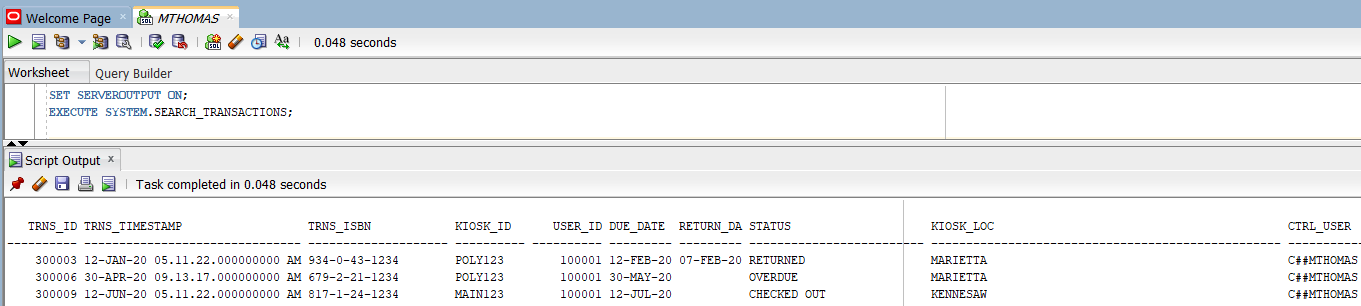
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_TRANSACTIONS;

Expected Results = Registered students only have access to view rows they generated on the table

Actual Results = User can only view their rows.

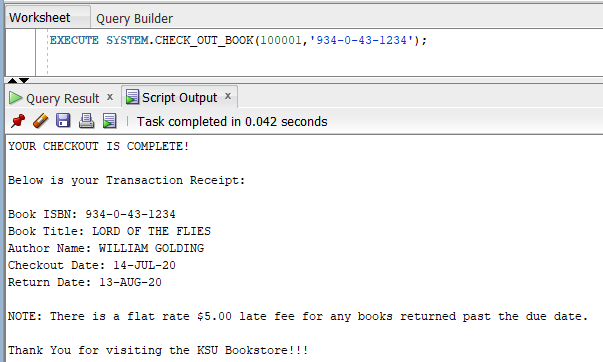


Scenario = User Checks Out a Book

Run Checkout Procedure

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.CHECK\_OUT\_BOOK(100001,'934-0-43-1234');



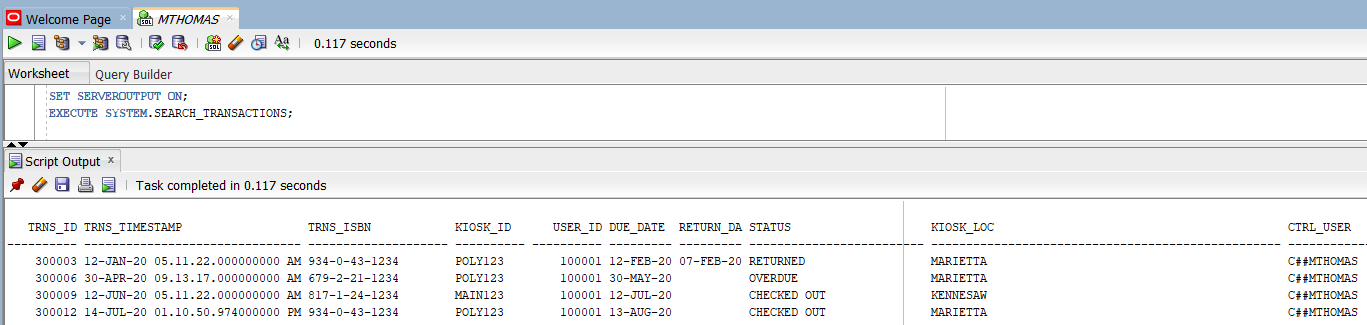
Rerun Procedure to Select Rows

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_TRANSACTIONS;

Expected Results = Registered students should see the NEW ROW from the recent checkout

Actual Results = User can only view their rows.



**Test Case #3:**

Scenario = User wants to view data on the TRANSACTIONS table

User Type = (Unregistered Student)

Username = C##SBASS

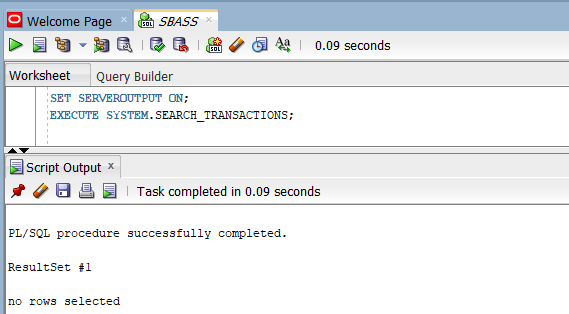
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_TRANSACTIONS;

Expected Results = Unregistered students DO NOT have access to view rows on the table

Actual Results = User cannot see rows



**USERS TABLE:**

**Test Case #1:**

Scenario = Employee wants to view all rows on USERS table

User Type = (Employee Non-Administrator)

Username = C##MLAMB

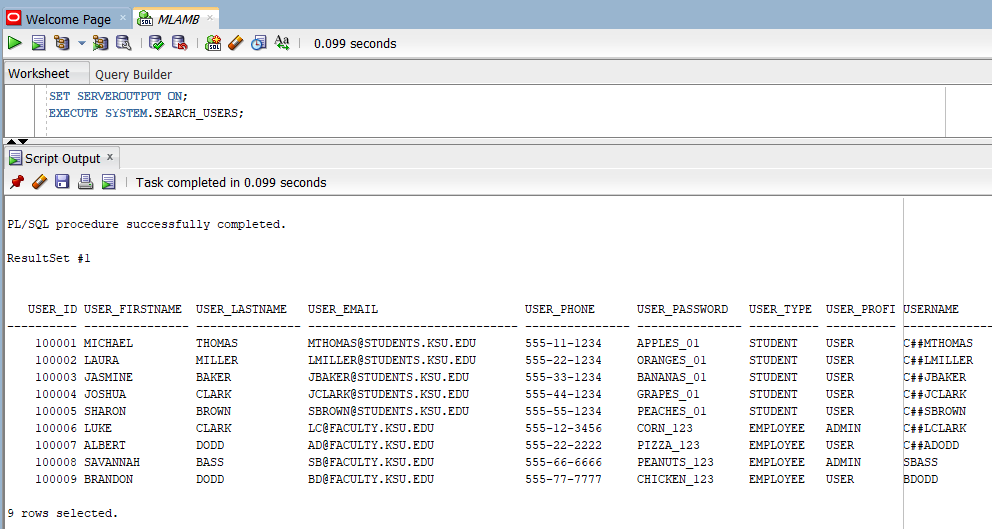
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_USERS;

Expected Result = Non-Administrator employees have access to view ALL rows on the table.

Actual Result = User can view all rows.



**Test Case #2:**

Scenario = User wants to view data on the USERS table

User Type = (Registered Student)

Username = C##LMILLER

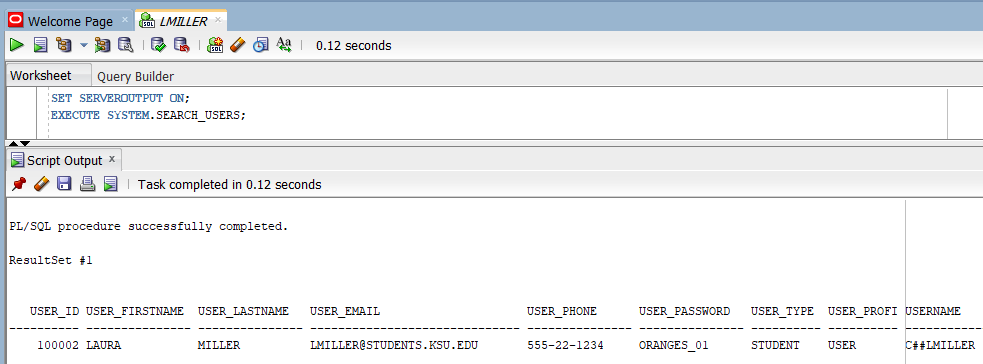
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_USERS;

Expected Results = Registered students only have access to view rows they generated on the table

Actual Results = User can only view their row



**Test Case #3:**

Scenario = User wants to view data on the USERS table

User Type = (Unregistered Student)

Username = C##SBASS

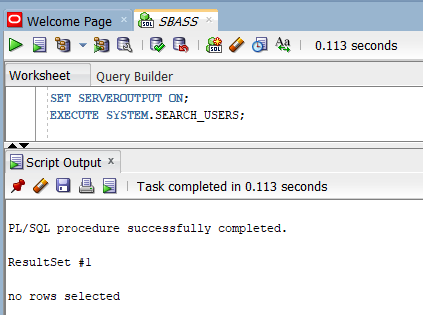
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_USERS;

Expected Results = Unregistered students DO NOT have access to view rows on the table

Actual Results = User cannot view any rows



**Note:** the VDP policy is enabled automatically.

### Virtual Private Database – DEMONSTRATION (Policy DISABLED)

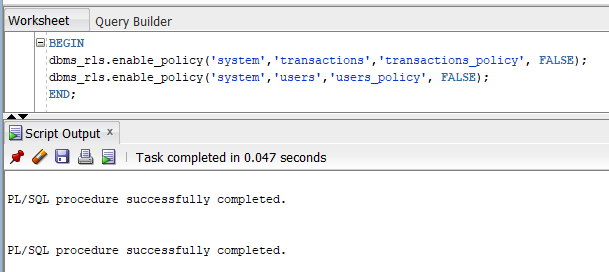
The following command was run to DISABLE the Policy:

BEGIN

dbms\_rls.enable\_policy('system','transactions','transactions\_policy', FALSE);

dbms\_rls.enable\_policy('system','users','users\_policy', FALSE);

END;



**Test Case #1: Rerun in Disabled mode**

Scenario = User wants to view data on the TRANSACTIONS table

User Type = (Registered Student)

Username = C##MTHOMAS

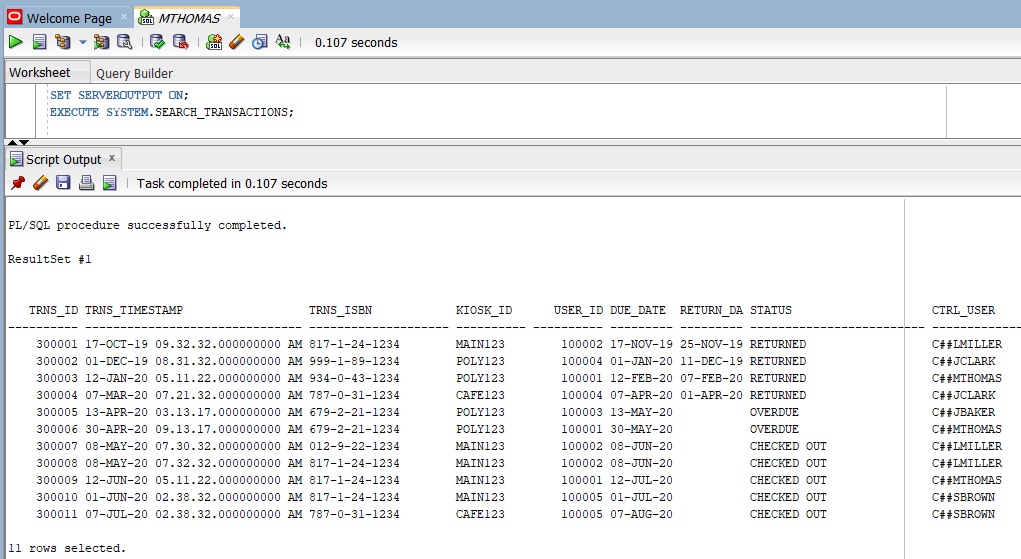
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_TRANSACTIONS;

Expected Results = Registered students should be able to view all rows on the table

Actual Results = User can view all rows.



**Test Case #2: Rerun in Disabled mode**

Scenario = User wants to view data on the USERS table

User Type = (Registered Student)

Username = C##LMILLER

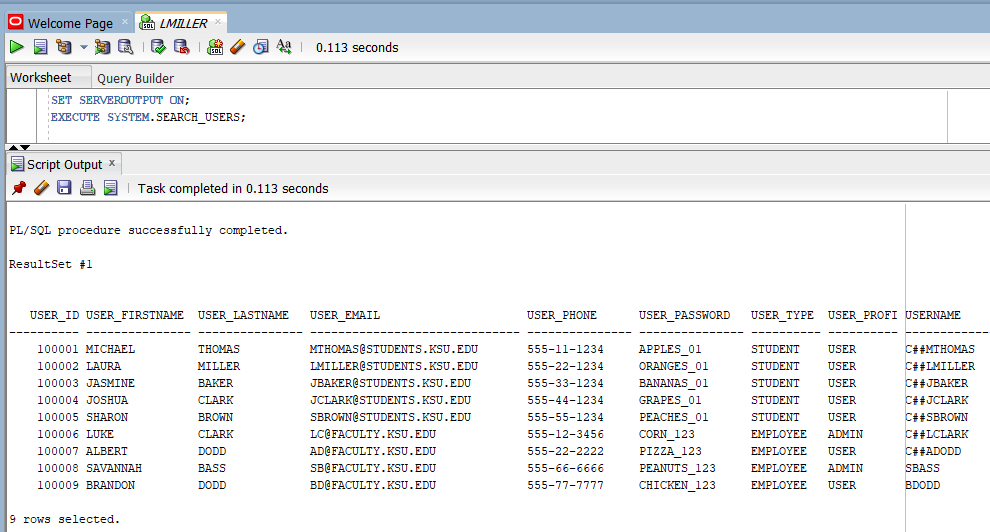
Run Procedure to Select

SET SERVEROUTPUT ON;

EXECUTE SYSTEM.SEARCH\_USERS;

Expected Results = Registered students should be able to view all rows on the table

Actual Results = User can view all rows



**Implementation of the Auditing Solution**

### Actions Audited by Default

Regardless of whether database auditing is enabled, Oracle Database *always* audits certain database-related operations and writes them to the operating system audit file. This fact is called mandatory auditing, and it includes the following operations:

* Connections to the instance with administrator privileges
* An audit record is generated that lists the operating system user connecting to Oracle Database as SYSOPER or SYSDBA. This provides for accountability of users with administrative privileges
* Database startup
* An audit record is generated that lists the operating system user starting the instance, the user terminal identifier, and the date and time stamp. This data is stored in the operating system audit trail because the database audit trail is not available until after the startup has successfully completed.
* Database shutdown
* An audit record is generated that lists the operating system user shutting down the instance, the user terminal identifier, and the date and time stamp.

There are a number of policies that are set up by default. Apart from that, we can also set Auditing for our Database according to our requirement. We have set up Unified Auditing model for our project. By default, Mixed mode auditing is enabled.

### Steps taken to enable Unified Auditing-

* Shut down oracle instance with command- shutdown immediate
* Stopped the listener with command- host lsnrctl stop
* Modified the orauniaud12.dll.dbl file to orauniaud12.dll file.
* Started the listener with command- host lsnrctl start
* Started Oracle instance- Startup
* Verified the status of unified auditing-

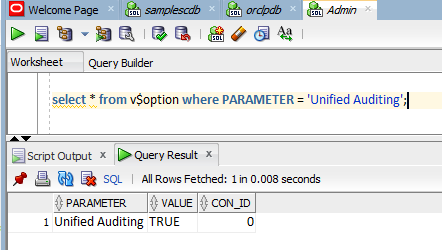
select \* from v$option where PARAMETER = 'Unified Auditing';

However, status was still FALSE

* Restarted the Oracle instance service and Oracle listener service
* Restarted the VM machine
* Verified the status of unified auditing-

select \* from v$option where PARAMETER = 'Unified Auditing';

Value was set to True.



**Details of the users in our Database-**

Admin- C##LCLARK

Students- C##SBROWN

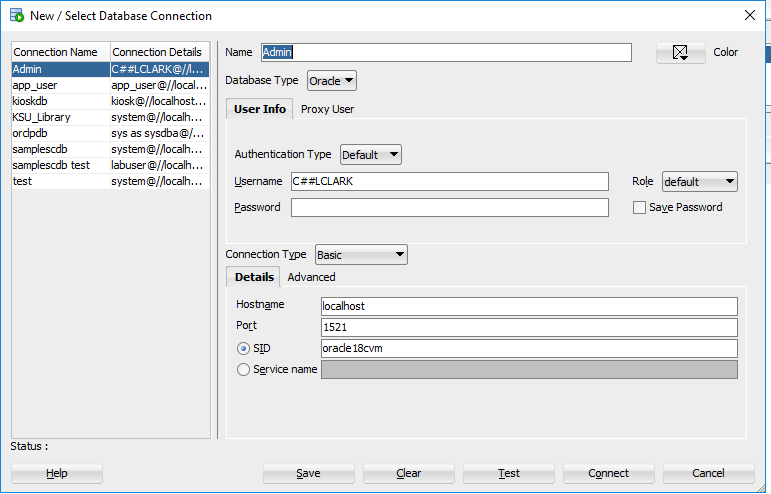
C##LMILLER

C##MTHOMAS

C##SBASS

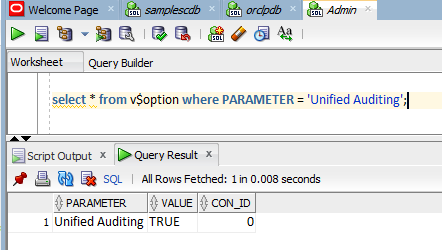
Employee with Limited Access- C##MLAMB

Logged in as C##LCLARK Admin account-



* Verify Unified Auditing Setting is Set to True.

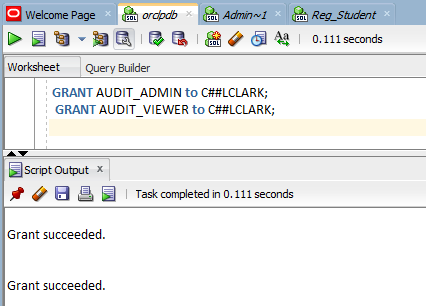
select \* from v$option where PARAMETER = 'Unified Auditing';



* Granted Audit\_Admin and Audit\_Viewer permissions to C##LCLARK

GRANT AUDIT\_ADMIN to C##LCLARK;

GRANT AUDIT\_VIEWER to C##LCLARK;



We have considered Unified auditing for below scenarios-

1. **Audit Failed logins-**

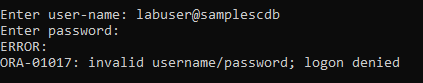
Connected as C##LCLARK admin login again and created Auditing Policy

The ORA\_LOGON\_FAILURES unified audit policy tracks failed logons

CREATE AUDIT POLICY ORA\_LOGON\_FAILURES ACTIONS LOGON;

Enable the Unified Auditing Policy-

AUDIT POLICY ORA\_LOGON\_FAILURES WHENEVER NOT SUCCESSFUL;



When unified auditing is enabled in Oracle Database, the audit records are populated in this new audit trail. This view displays audit records in tabular form by retrieving the audit records from the audit trail as shows in below screenshot. Below code will display the failed logins which happened in last 10 minutes.

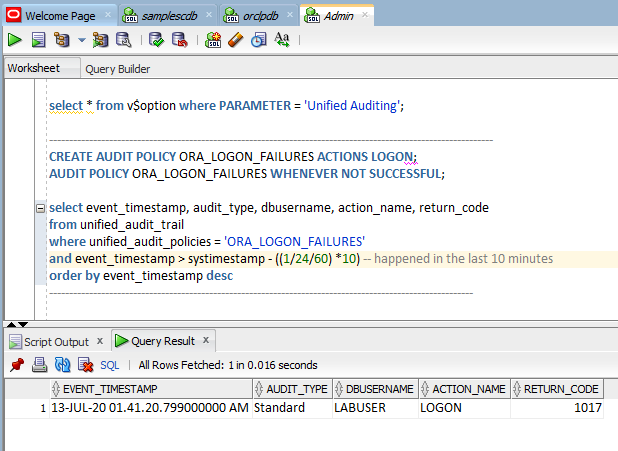
select event\_timestamp, audit\_type, dbusername, action\_name, return\_code

from unified\_audit\_trail

where unified\_audit\_policies = 'ORA\_LOGON\_FAILURES'

and event\_timestamp > systimestamp - ((1/24/60) \*10) -- happened in the last 10 minutes

order by event\_timestamp desc



1. **Audit Privileges-**

Registered EMPLOYEE C##MLAMB can select SELECT, INSERT, UPDATE and DELETE into any table however, EMPLOYEE C##MLAMB doesn’t have permission to create, alter or drop any table. In case, a user tries to create, alter or drop any table. Logs will be saved in unified\_Audit\_trail for tracking.

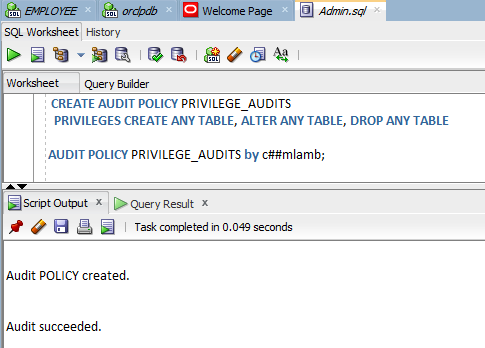
* Audit Policy created-

CREATE AUDIT POLICY PRIVILEGE\_AUDITS

PRIVILEGES CREATE ANY TABLE, ALTER ANY TABLE, DROP ANY TABLE

* Enabled the Audit policy-

AUDIT POLICY PRIVILEGE\_AUDITS by c##mlamb;



* Audit Policy Created Successfully-

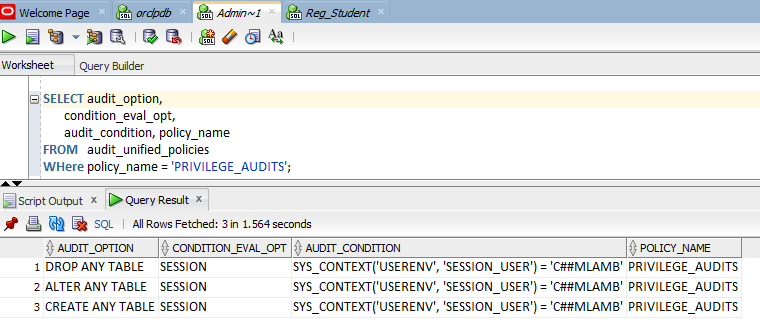
SELECT audit\_option,

condition\_eval\_opt,

audit\_condition, policy\_name

FROM audit\_unified\_policies

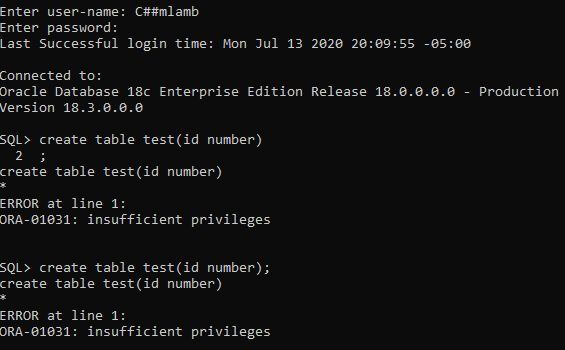
WHere policy\_name = 'PRIVILEGE\_AUDITS';



**Test Case-**

* Connected as c##MLAMB
* Tried to create a table
* Since, we have insufficient privileges, we couldn’t create a table. Since Admin C##LCLARK will be able to view in audits if an employee c##mlamb will try to create a table.

Create table test(id number);



* **Check Audit Trails-** Logs will be saved in unified\_Audit\_trail to track.

SELECT event\_timestamp,

dbusername,

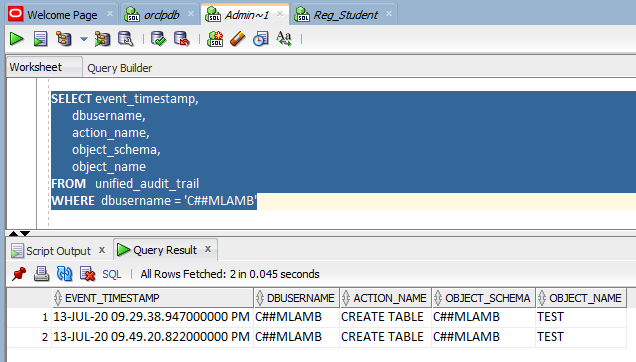
action\_name,

object\_schema,

object\_name

FROM unified\_audit\_trail

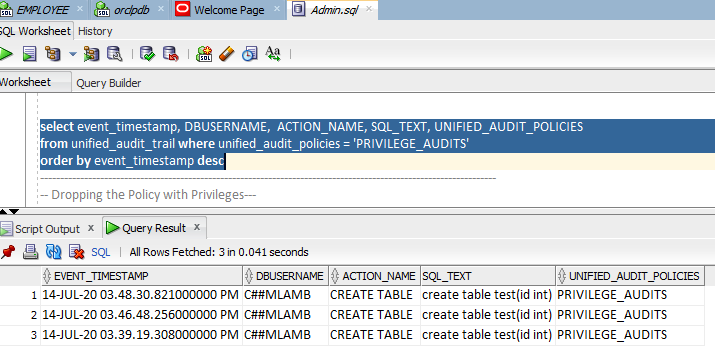
WHERE dbusername = 'C##MLAMB'



select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

from unified\_audit\_trail where unified\_audit\_policies = 'PRIVILEGE\_AUDITS'

order by event\_timestamp desc



1. **Auditing Searching Books**

Here we are auditing select statement on Books table. For example- When a student tries to search a book. Audits will be created in Unified\_audit\_trail.

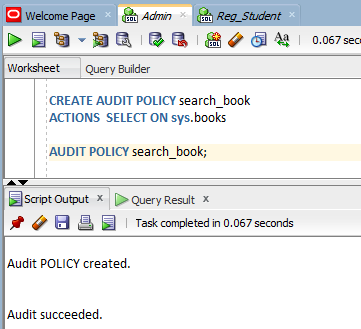
* Audit policy created-

CREATE AUDIT POLICY search\_book

ACTIONS SELECT ON sys.books

* Enable the Audit policy-

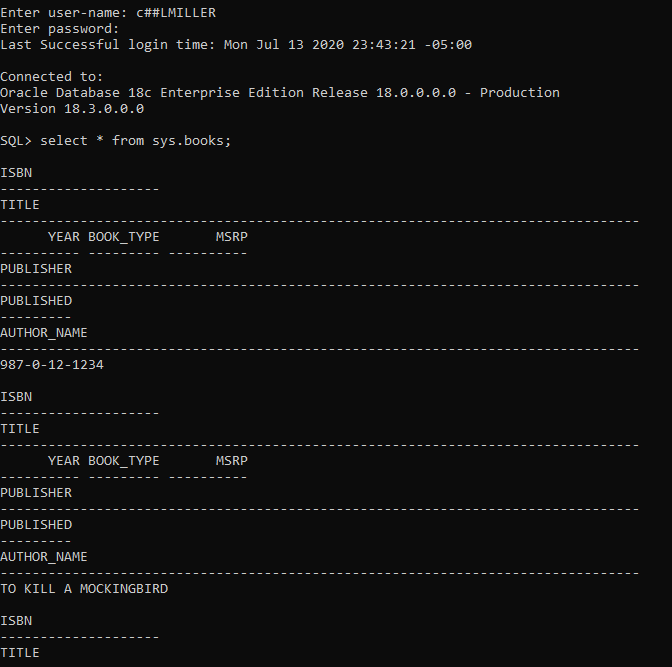
AUDIT POLICY search\_book;



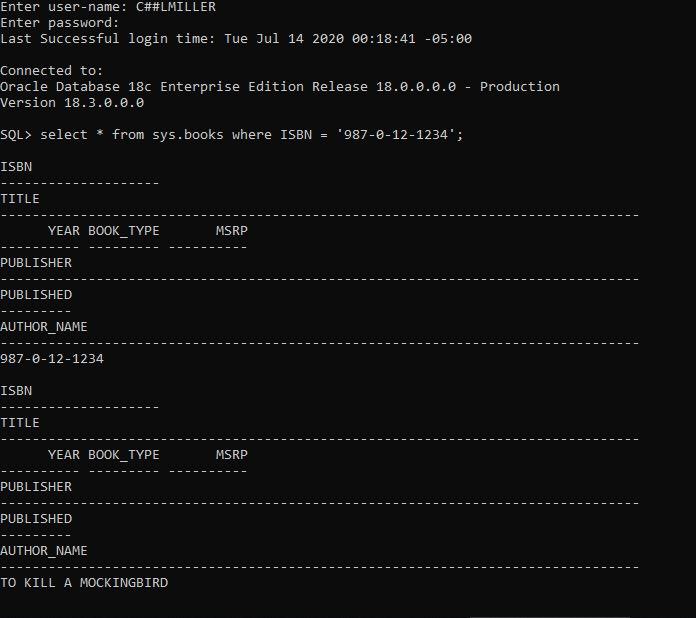
**Test Case**- Lets perform the query with two different users.

* Logged in with Registered student account C##LMILLER
* Try to search for the book

Select \* from sys.books;



select \* from sys.books where ISBN = '987-0-12-1234';



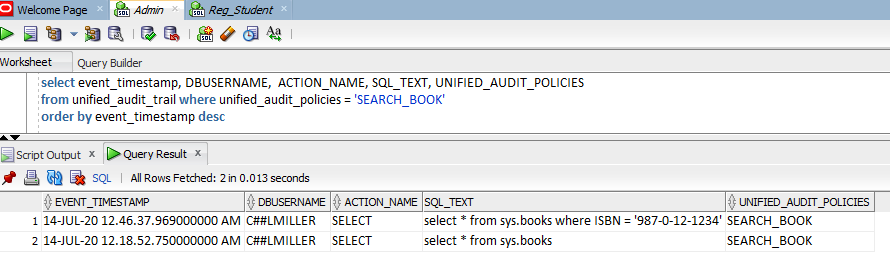
* **Check Audit Trails-**

This SELECT statement will cause unified audit records to be generated. You can look at these audit records from the UNIFIED\_AUDIT\_TRAIL view by using the query shown

select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

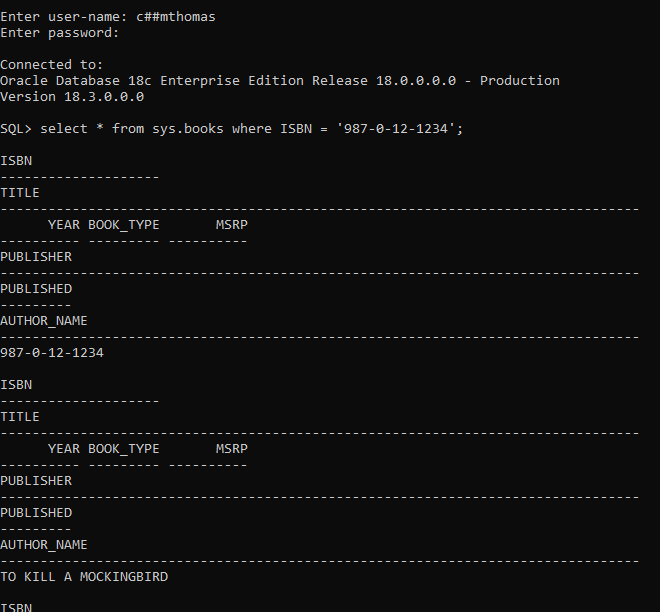
from unified\_audit\_trail where unified\_audit\_policies = 'SEARCH\_BOOK'

order by event\_timestamp desc

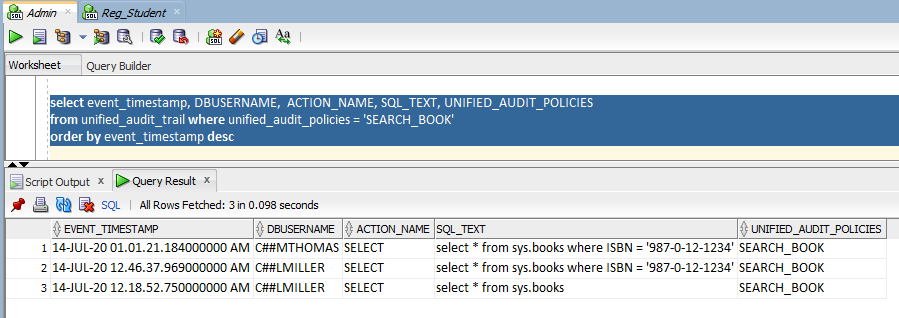


* Connected as registered user C##MTHOMAS

select \* from sys.books where ISBN = '987-0-12-1234';



* **Check Unified Audit Trail-** Unified Audit records are generated for C##MTHOMAS and C##LMILLER.



1. **Audit Updates on Kiosks Table-**

We can audit updates made on Kiosks table for example- When Admin updates the book information in Kiosks table that data will be recorded in unified audit trail for tracking purpose in future.

* **Policy Created-**

create audit policy Kiosks\_Audit

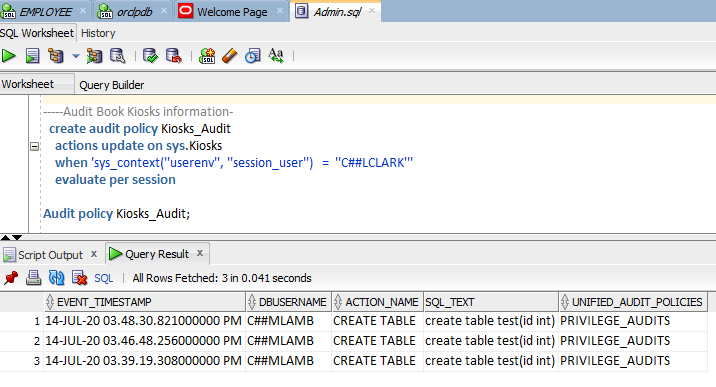
actions update on sys.Kiosks

when 'sys\_context(''userenv'', ''session\_user'') = ''C##LCLARK'''

evaluate per session

* **Enabled the policy**

Audit policy Kiosks\_Audit;



* **Verify the policy is created-**

SELECT audit\_option,

condition\_eval\_opt,

audit\_condition, policy\_name

FROM audit\_unified\_policies

WHere policy\_name = 'Kiosks\_Audit';

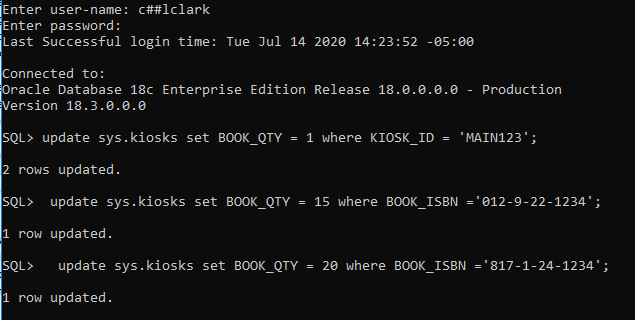
**Test Case-**

* Logged in as Admin C##LCLARK

--- update sys.kiosks set BOOK\_QTY = 1 where KIOSK\_ID = 'MAIN123';

update sys.kiosks set BOOK\_QTY = 15 where BOOK\_ISBN ='012-9-22-1234';

update sys.kiosks set BOOK\_QTY = 20 where BOOK\_ISBN ='817-1-24-1234';



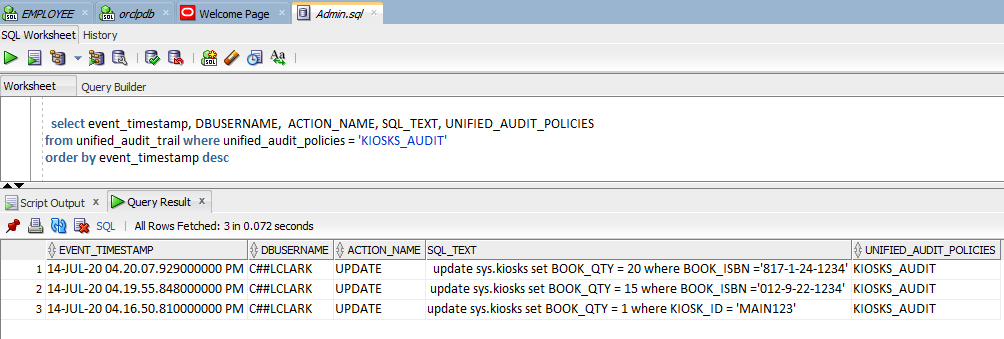
* **Check the Unified Audit Trail-**

We can verify that all the update records are tracked in unified audit trail. Admin can track when the Kiosks book information was updated.

select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

from unified\_audit\_trail where unified\_audit\_policies = 'KIOSKS\_AUDIT'

order by event\_timestamp desc



1. **Audit Actions on Transaction table-**

Now, we will audit all action made by students/users in Transaction table. Whenever a student tries to select, insert, update or delete the data, Unified audits will be recorded.

Please note that Student only has access to select and insert. Users doesn’t have access to delete or update in Transaction table. Still, if a user tries to delete or update in Transactions table then the unified trail will be recorded. That’s who we can Audit all the actions performed by user.

create audit policy Transactions\_audit

actions all on sys.TRANSACTIONS

when 'sys\_context(''userenv'', ''session\_user'') in (''C##SBASS'', ''C##MTHOMAS'', ''C##LMILLER'', ''C##SBROWN'')'

evaluate per session

Policy created can be seen by below command-

SELECT audit\_option,

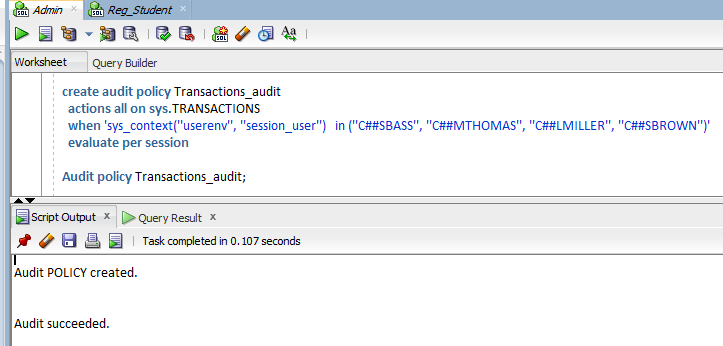
condition\_eval\_opt,

audit\_condition, policy\_name

FROM audit\_unified\_policies

WHere policy\_name = 'TRANSACTIONS\_AUDIT';

Audit policy Transactions\_audit;



**Test Case-**

* Connected as C##MTHOMAS (one of the registered student)
* Tried to Select, Insert, Update and Delete in Transactions table.

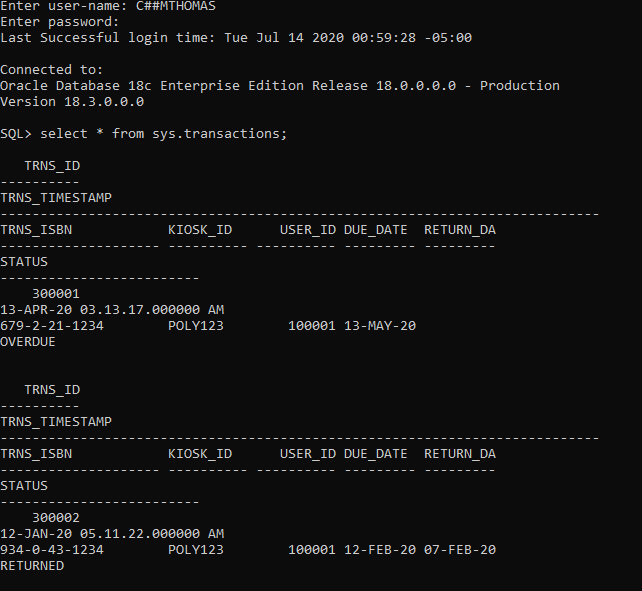
Select \* from sys.transactions;

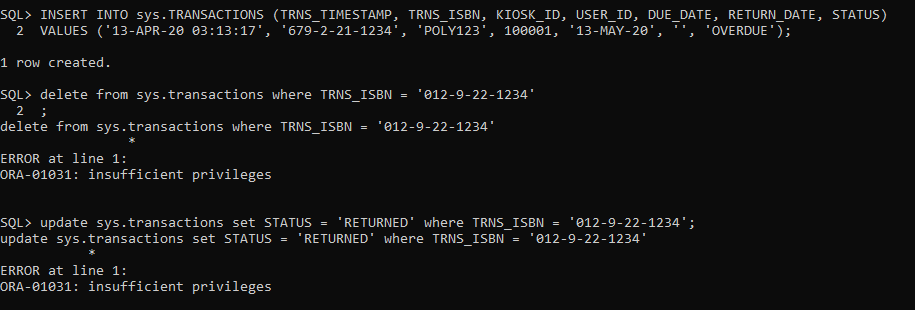
INSERT INTO sys.TRANSACTIONS (TRNS\_TIMESTAMP, TRNS\_ISBN, KIOSK\_ID, USER\_ID, DUE\_DATE, RETURN\_DATE, STATUS)

VALUES ('13-APR-20 03:13:17', '679-2-21-1234', 'POLY123', 100001, '13-MAY-20', '', 'OVERDUE');

delete from sys.transactions where TRNS\_ISBN = '012-9-22-1234'

update sys.transactions set STATUS = 'RETURNED' where TRNS\_ISBN = '012-9-22-1234'





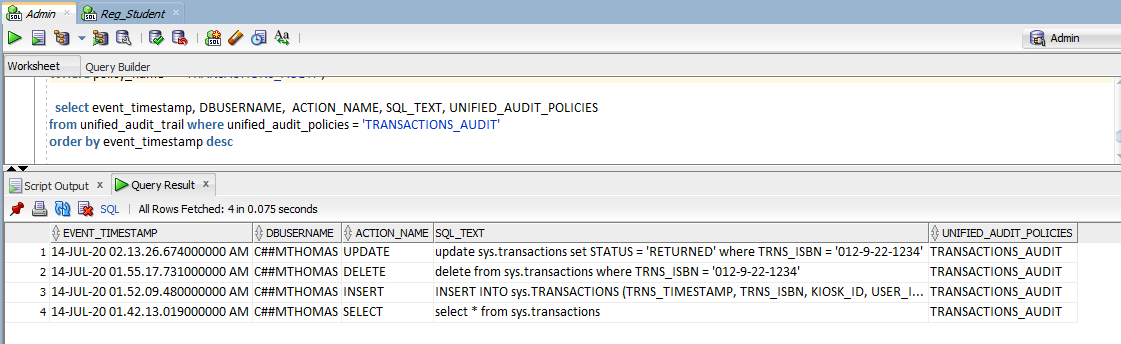
* **Check Unified Audit Trail-**

For user C##MTHOMAS user, Select, Insert, Update and Delete Action on Transactions table are recorded in Unified Auditing trail

select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

from unified\_audit\_trail where unified\_audit\_policies = 'TRANSACTIONS\_AUDIT'

order by event\_timestamp desc



1. **Audit Users deletion-**

When an Admin try to delete the Employee who is no longer with the company then those records will be generated to unified Audit trail for tracking.

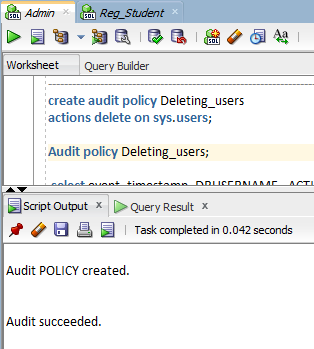
* **Created Policy-**

create audit policy Deleting\_users

actions delete on sys.users;

* **Enabled the Policy**

Audit policy Deleting\_users;



* **Verify the policy is created-**

SELECT audit\_option,

condition\_eval\_opt,

audit\_condition, policy\_name

FROM audit\_unified\_policies

WHere policy\_name = 'DELETING\_USERS';

**Test Case-**

* Connected as Admin user
* Tried to delete the Employees who are not with the company right now.

delete from sys.users where USER\_FIRSTNAME = 'SANDRA' and USER\_TYPE = 'EMPLOYEE'

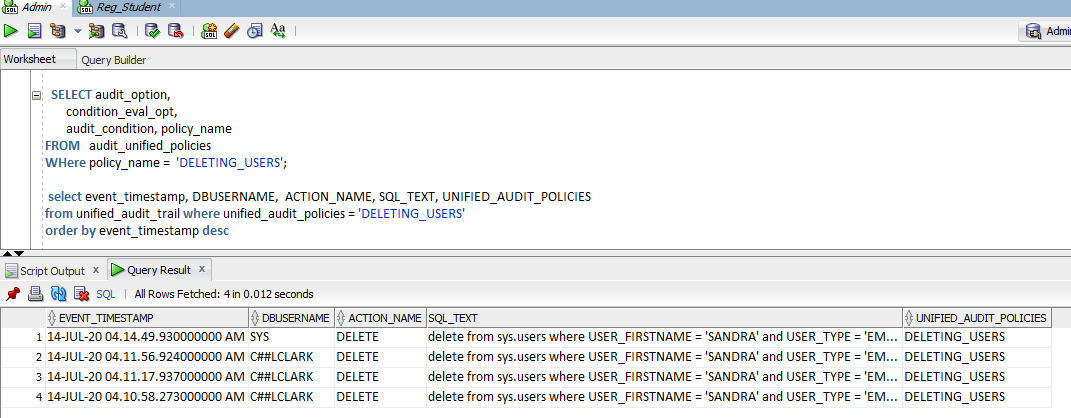


* **Check the unified Audit Trail-** Whenever someone will try to delete the user then record will be updated in unified auditing trail.

select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

from unified\_audit\_trail where unified\_audit\_policies = 'DELETING\_USERS'

order by event\_timestamp desc



**We can Drop the Unified Audit policy-**

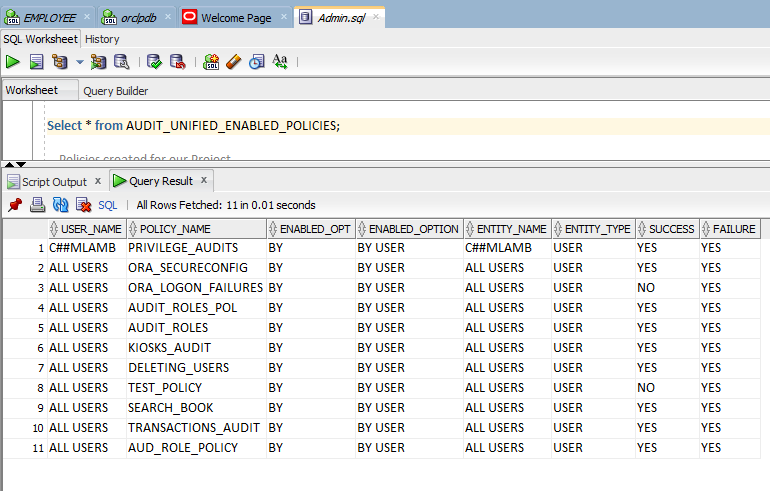
In order to drop the audit policy, first we have set it to NoAudit mode then we can drop any policy

NOAUDIT POLICY DELETING\_USERS;

DROP AUDIT POLICY DELETING\_USERS;

**View the Enabled Unified Audit policy-**

Select \* from AUDIT\_UNIFIED\_ENABLED\_POLICIES;

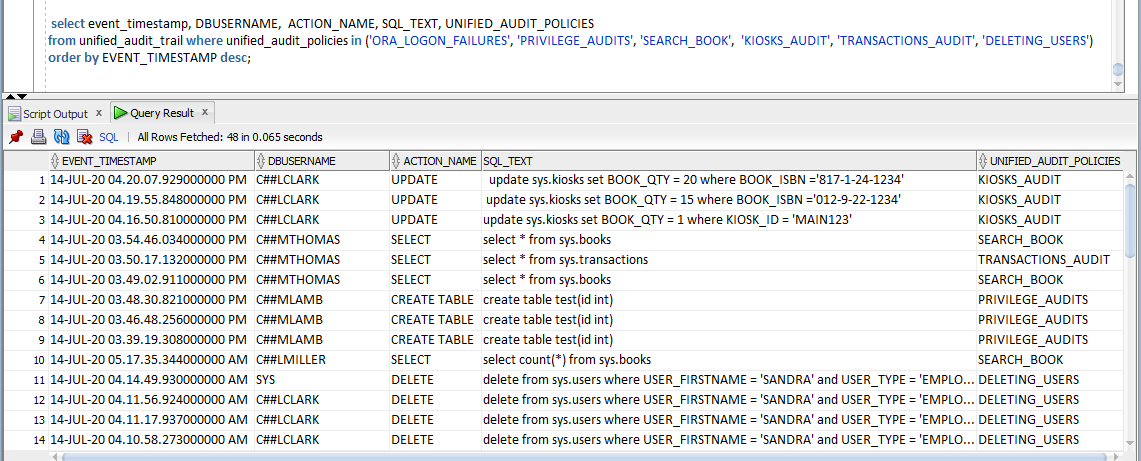


**Unified Policies Created for our Project-**

select event\_timestamp, DBUSERNAME, ACTION\_NAME, SQL\_TEXT, UNIFIED\_AUDIT\_POLICIES

from unified\_audit\_trail where unified\_audit\_policies in ('ORA\_LOGON\_FAILURES', 'PRIVILEGE\_AUDITS', 'SEARCH\_BOOK', 'KIOSKS\_AUDIT', 'TRANSACTIONS\_AUDIT', 'DELETING\_USERS')

order by EVENT\_TIMESTAMP desc;



Thank you!