Certainly! Below is the **complete PL/SQL-compatible version** of your **Role-Based Access Control (RBAC)** system designed to run on **Oracle Live SQL**. It includes:

Table creation with PK–FK  
Sample data insertion  
Procedure with role-based access control  
Simulation of access allowed or denied based on role

**Step 1: Create ROLES and USERS Tables**

-- Drop if exists for re-runs

DROP TABLE Users CASCADE CONSTRAINTS;

DROP TABLE Roles CASCADE CONSTRAINTS;

-- Create Roles table

CREATE TABLE Roles (

role\_id NUMBER PRIMARY KEY,

role\_name VARCHAR2(20) UNIQUE

);

-- Create Users table with foreign key to Roles

CREATE TABLE Users (

user\_id NUMBER PRIMARY KEY,

username VARCHAR2(50) UNIQUE,

role\_id NUMBER,

CONSTRAINT fk\_role FOREIGN KEY (role\_id) REFERENCES Roles(role\_id)

);

**Step 2: Insert Sample Data**

-- Insert Roles

INSERT INTO Roles (role\_id, role\_name) VALUES (1, 'Admin');

INSERT INTO Roles (role\_id, role\_name) VALUES (2, 'Teller');

INSERT INTO Roles (role\_id, role\_name) VALUES (3, 'Auditor');

-- Insert Users

INSERT INTO Users (user\_id, username, role\_id) VALUES (101, 'amit\_admin', 1);

INSERT INTO Users (user\_id, username, role\_id) VALUES (102, 'beenu\_teller', 2);

INSERT INTO Users (user\_id, username, role\_id) VALUES (103, 'chandan\_auditor', 3);

COMMIT;

**Step 3: Create Procedure to Simulate Grant Logic**

CREATE OR REPLACE PROCEDURE CheckAccess (p\_user\_id IN NUMBER) IS

v\_role VARCHAR2(20);

BEGIN

-- Fetch role name for the given user ID

SELECT r.role\_name INTO v\_role

FROM Users u

JOIN Roles r ON u.role\_id = r.role\_id

WHERE u.user\_id = p\_user\_id;

-- Check if Admin

IF v\_role != 'Admin' THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Access Denied: Admin role required');

ELSE

DBMS\_OUTPUT.PUT\_LINE(' Access Granted. Welcome Admin!');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(' User not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Nope' || SQLERRM);

END;

/

**Step 4: Run Simulations**

**Admin (Allowed)**

BEGIN

CheckAccess(101); -- amit\_admin

END;

/

Output:

Access Granted. Welcome Admin!

**Teller (Access Denied)**

BEGIN

CheckAccess(102); -- beenu\_teller

END;

/

Output:

ORA-20001: Access Denied: Admin role required

**Auditor (Access Denied)**

BEGIN

CheckAccess(103); -- chandan\_auditor

END;

/

Output:

ORA-20001: Access Denied: Admin role required

## Optional Enhancement: More Role-Specific Logic

IF v\_role = 'Teller' THEN

DBMS\_OUTPUT.PUT\_LINE('Access granted to Teller functions.');

ELSIF v\_role = 'Auditor' THEN

DBMS\_OUTPUT.PUT\_LINE('Access granted to view-only functions.');

ELSIF v\_role = 'Admin' THEN

DBMS\_OUTPUT.PUT\_LINE('Access granted to all admin functions.');

ELSE

RAISE\_APPLICATION\_ERROR(-20001, 'Access Denied.');

END IF;

Would you like a **bonus case** where access is checked against multiple privileges (e.g., loans, accounts)?

different banking functions like **Accounts** and **Loans**, based on user **roles** in **Oracle Live SQL**.

Bank roles:

* **Admin** → Full access
* **Teller** → Only account operations
* **LoanOfficer** → Only loan operations
* **Auditor** → Read-only view

## 1. Add LoanOfficer Role

First, make sure the extra role exists:

INSERT INTO Roles (role\_id, role\_name) VALUES (4, 'LoanOfficer');

-- Add a user with that role

INSERT INTO Users (user\_id, username, role\_id) VALUES (104, 'david\_loan', 4);

COMMIT;

2. Enhanced Procedure: CheckAccessWithPrivileges

CREATE OR REPLACE PROCEDURE CheckAccessWithPrivileges (

p\_user\_id IN NUMBER,

p\_function IN VARCHAR2 -- 'ACCOUNTS', 'LOANS', 'REPORTS'

) IS

v\_role VARCHAR2(20);

BEGIN

-- Fetch role of the user

SELECT r.role\_name INTO v\_role

FROM Users u

JOIN Roles r ON u.role\_id = r.role\_id

WHERE u.user\_id = p\_user\_id;

-- Simulate access logic

IF v\_role = 'Admin' THEN

DBMS\_OUTPUT.PUT\_LINE(' Admin: Access granted to all functions.');

ELSIF v\_role = 'Teller' AND p\_function = 'ACCOUNTS' THEN

DBMS\_OUTPUT.PUT\_LINE(' Teller: Access granted to Accounts.');

ELSIF v\_role = 'LoanOfficer' AND p\_function = 'LOANS' THEN

DBMS\_OUTPUT.PUT\_LINE(' Loan Officer: Access granted to Loans.');

ELSIF v\_role = 'Auditor' AND p\_function IN ('ACCOUNTS', 'LOANS', 'REPORTS') THEN

DBMS\_OUTPUT.PUT\_LINE(' Auditor: Read-only access granted.');

ELSE

RAISE\_APPLICATION\_ERROR(-20001, ' Access Denied: You are not allowed to access ' || p\_function || '.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(' User not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Nope ' || SQLERRM);

END;

/

## 3. Run Simulations

### a. Admin accessing Loans

BEGIN

CheckAccessWithPrivileges(101, 'LOANS');

END;

/

Output:

Admin: Access granted to all functions.

### b. Teller accessing Accounts

BEGIN

CheckAccessWithPrivileges(102, 'ACCOUNTS');

END;

/

Output:

Teller: Access granted to Accounts.

### c. Teller accessing Loans

BEGIN

CheckAccessWithPrivileges(102, 'LOANS');

END;

/

Output:

ORA-20001: ❌ Access Denied: You are not allowed to access LOANS.

### d. Auditor accessing Reports

BEGIN

CheckAccessWithPrivileges(103, 'REPORTS');

END;

/

Output:

Auditor: Read-only access granted.

### e. Loan Officer accessing Loans

BEGIN

CheckAccessWithPrivileges(104, 'LOANS');

END;

/

Output:

Loan Officer: Access granted to Loans.

## Summary Table

| **Role** | **Can Access** |
| --- | --- |
| Admin | All |
| Teller | ACCOUNTS only |
| LoanOfficer | LOANS only |
| Auditor | View only (Accounts, Loans, Reports) |