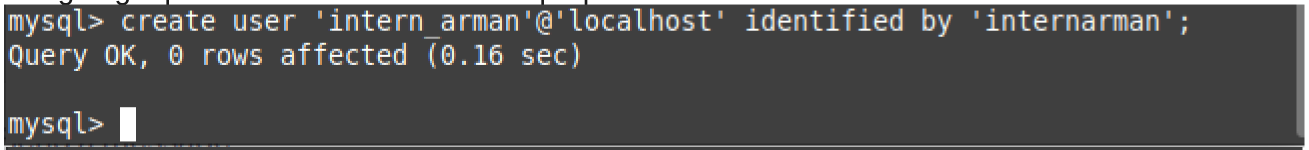
Database User management

Last updated by | shakibesaib | Aug 15, 2023 at 4:52 PM GMT+6

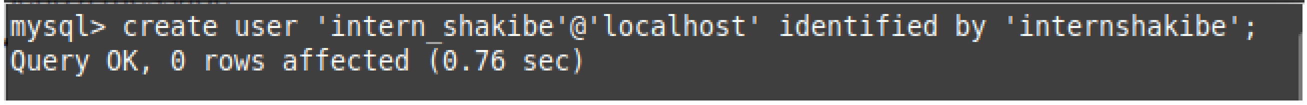
Database user management involves the administration and control of user access to a database.

Here are the key aspects of managing database users:

- **User Creation:** Create new users in the database. This typically involves providing a username and assigning a password for authentication purposes.

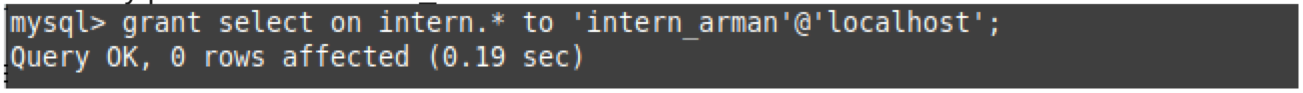


```
mysql> create user 'intern_arman'@'localhost' identified by 'internarman';
Query OK, 0 rows affected (0.16 sec)

mysql>
```



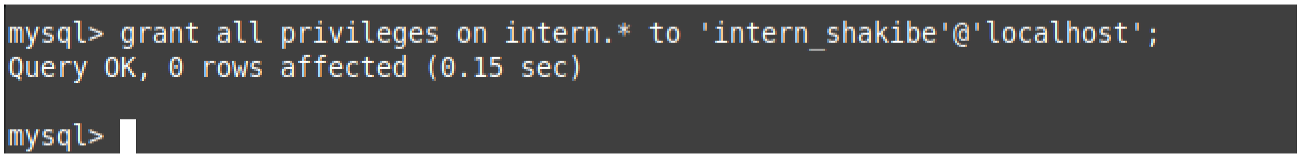
```
mysql> create user 'intern_shakibe'@'localhost' identified by 'internshakibe';
Query OK, 0 rows affected (0.76 sec)
```

- **User Permissions:** Assign appropriate permissions to users based on their roles and responsibilities. Permissions control what actions a user can perform on the database, such as read, write, update, or delete data.
  Read only permission for intern_arman



```
mysql> grant select on intern.* to 'intern_arman'@'localhost';
Query OK, 0 rows affected (0.19 sec)
```

GRANT ALL PRIVILEGES assigns all privileges (full access) to the user 'intern_shakibe' on the 'intern' database.



```
mysql> grant all privileges on intern.* to 'intern_shakibe'@'localhost';
Query OK, 0 rows affected (0.15 sec)

mysql>
```

- **User Roles:** Define roles or groups with predefined sets of permissions. Assign users to these roles to simplify permission management and ensure consistent access control.
  Creating Roles:

```
CREATE ROLE readonly;
CREATE ROLE readwrite;
CREATE ROLE supervisor;
```

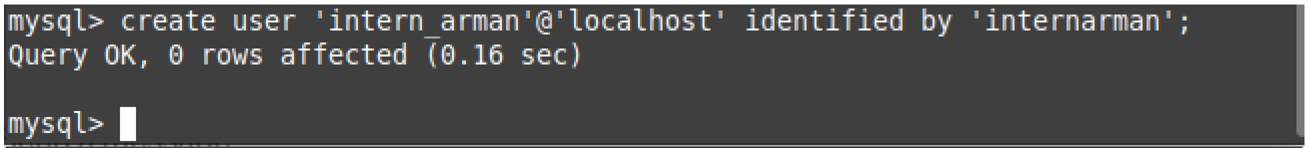
Granting permission for Roles:

```
GRANT SELECT ON intern.* TO readonly;
GRANT SELECT, INSERT, UPDATE ON intern.* TO readwrite;
```

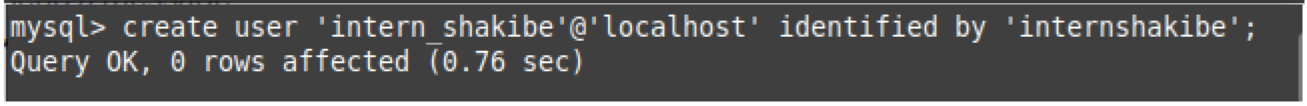
Assigning users to Roles:

```
GRANT readonly TO 'intern_arman'@'localhost';
GRANT readwrite TO 'intern_shakibe'@'localhost';
```

- **User Authentication:** Establish a mechanism to verify the identity of users before granting access to the database. This can include methods like username/password authentication, integration with an external authentication system, or the use of digital certificates.

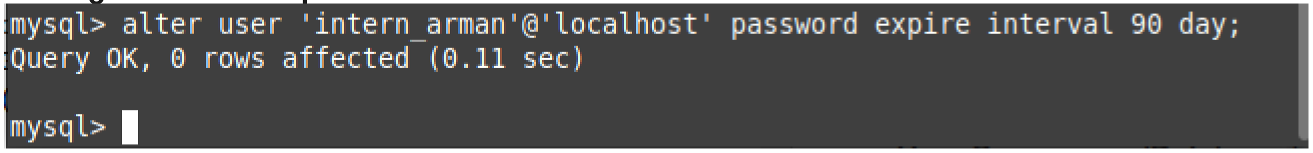


```
mysql> create user 'intern_arman'@'localhost' identified by 'internarman';
Query OK, 0 rows affected (0.16 sec)

mysql>
```



```
mysql> create user 'intern_shakibe'@'localhost' identified by 'internshakibe';
Query OK, 0 rows affected (0.76 sec)
```

- **Password Management:** Implement policies and guidelines for password creation, expiration, and complexity. Enforce password changes periodically to enhance security.
  Setting Password Expiration:



```
mysql> alter user 'intern_arman'@'localhost' password expire interval 90 day;
Query OK, 0 rows affected (0.11 sec)

mysql>
```

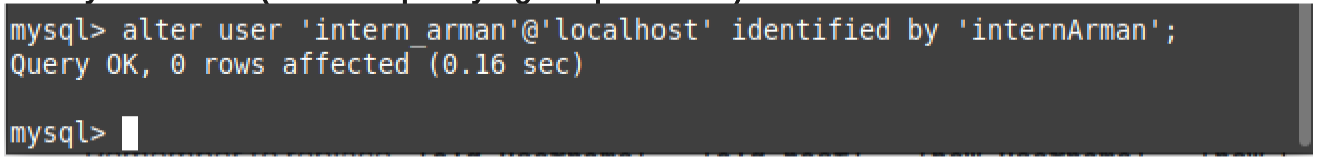
  This will expire the password after 90 days.
  Enforce Password Complexity: Enforce password complexity rules to ensure that passwords are strong and not easily guessable. MySQL has a built-in function validate_password for this purpose.

```
SHOW VARIABLES LIKE 'validate_password%';
SET GLOBAL validate_password.policy=LOW;
```

- **User Modification:** Allow administrators to modify user information such as username, password, and contact details when necessary. This may involve changing permissions or roles assigned to the user.
  Revoke existing privileges:

```
REVOKE ALL PRIVILEGES ON *.* FROM 'intern_arman'@'localhost';
```

  Modify Password (without specifying old password)



```
mysql> alter user 'intern_arman'@'localhost' identified by 'internArman';
Query OK, 0 rows affected (0.16 sec)

mysql>
```

- **User Suspension/Deletion:** Temporarily suspend or permanently delete user accounts when they are no longer needed or when access needs to be revoked. This helps maintain data security and privacy.
  User Deletion:

```
-- Delete user account
DROP USER 'intern_arman'@'localhost';

-- Flush privileges to apply changes
FLUSH PRIVILEGES;
```

- **User Auditing:** Monitor user activity within the database. Keep track of login attempts, queries executed, and modifications made by users. Audit logs can be used for troubleshooting, security analysis, and compliance purposes.
  For this, we need to install MySQL Audit Plugin. Then we need to modify the mysql configuration file.

```
[mysqld]
plugin-load = audit_log.so
audit_log = FORCE_PLUS_PERMANENT
audit_log_format = JSON
audit_log_policy = ALL
```

This configuration enables the MySQL Audit Plugin, specifies the audit log format as JSON, and sets the audit policy to log all events.
View user logins, queries, and modifications:

```
SELECT * FROM mysql.audit_log WHERE event_time >= '2023-01-01';
```

This query fetches audit logs from January 1, 2023, onwards.

- **User Training and Support:** Provide training and support to users on database access, security best practices, and proper use of database tools. This ensures users understand their responsibilities and helps prevent accidental or unauthorized access.

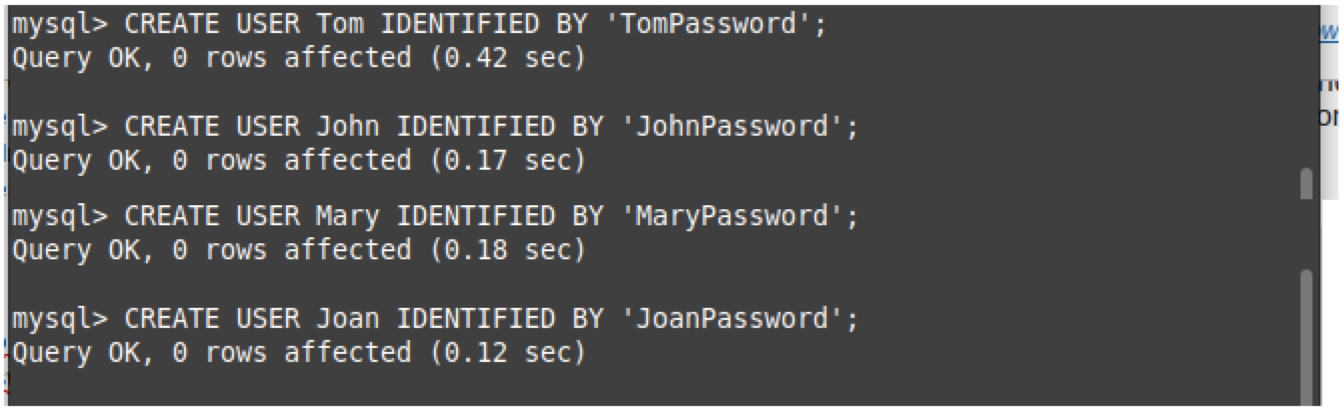
- **User Documentation:** Maintain proper documentation that outlines the user management processes, permissions, and roles. This documentation serves as a reference for administrators and helps ensure consistent user management practices.

Implementation of Users, Groups, Roles, and Permissions

- **Creating Database:**
  ```
  {CREATE DATABASE School;}
  ```

- **Creating User:**

```
{Use School;
CREATE USER Tom WITH PASSWORD = 'Tompassword';
CREATE USER John WITH PASSWORD = 'Johnpassword';
CREATE USER Mary WITH PASSWORD = 'Marypassword';
CREATE USER Joan WITH PASSWORD = 'Joanpassword';
CREATE USER Tom IDENTIFIED BY 'TomPassword';}
```

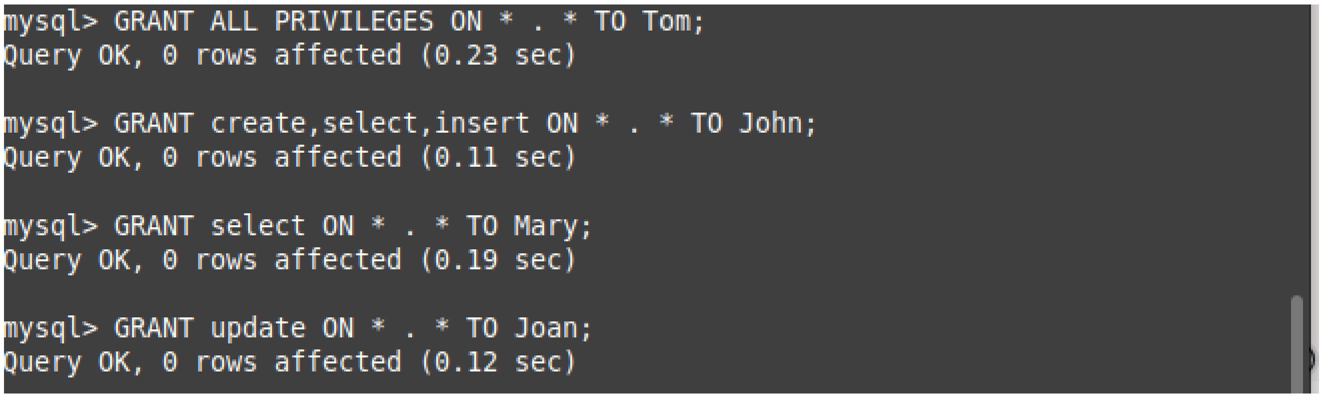


```
mysql> CREATE USER Tom IDENTIFIED BY 'TomPassword';
Query OK, 0 rows affected (0.42 sec)

mysql> CREATE USER John IDENTIFIED BY 'JohnPassword';
Query OK, 0 rows affected (0.17 sec)

mysql> CREATE USER Mary IDENTIFIED BY 'MaryPassword';
Query OK, 0 rows affected (0.18 sec)

mysql> CREATE USER Joan IDENTIFIED BY 'JoanPassword';
Query OK, 0 rows affected (0.12 sec)
```

- **Granting various permissions to the database users:**

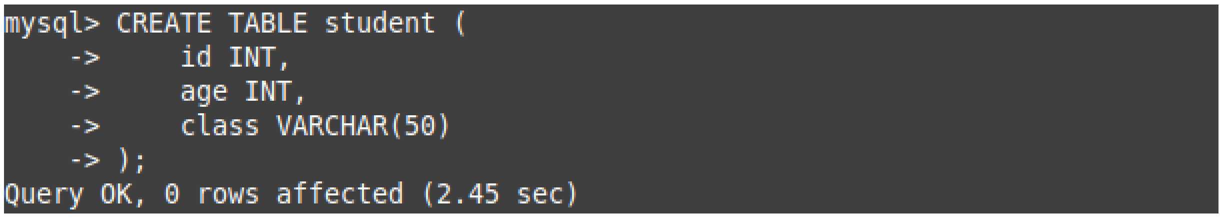
1. ALL PRIVILEGES: It permits all privileges to a new user account.
2. CREATE: It enables the user account to create databases and tables.
3. DROP: It enables the user account to drop databases and tables.
4. DELETE: It enables the user account to delete rows from a specific table.
5. INSERT: It enables the user account to insert rows into a specific table.
6. SELECT: It enables the user account to read a database.
7. UPDATE: It enables the user account to update table rows.

```
GRANT ALL PRIVILEGES ON * . * TO Tom;
GRANT create,select,insert ON * . * TO John;
GRANT select ON * . * TO Mary;
GRANT update ON * . * TO Joan;
```

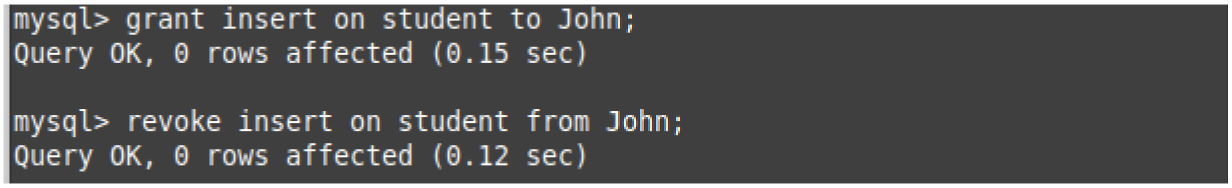


```
mysql> GRANT ALL PRIVILEGES ON * . * TO Tom;
Query OK, 0 rows affected (0.23 sec)

mysql> GRANT create,select,insert ON * . * TO John;
Query OK, 0 rows affected (0.11 sec)

mysql> GRANT select ON * . * TO Mary;
Query OK, 0 rows affected (0.19 sec)

mysql> GRANT update ON * . * TO Joan;
Query OK, 0 rows affected (0.12 sec)
```

- **Revoking permissions is the converse of granting permissions on database objects.**
  `revoke insert on student from John;`
  here **student** is a table.



```
mysql> CREATE TABLE student (
    ->     id INT,
    ->     age INT,
    ->     class VARCHAR(50)
    -> );
Query OK, 0 rows affected (2.45 sec)
```



```
mysql> grant insert on student to John;
Query OK, 0 rows affected (0.15 sec)

mysql> revoke insert on student from John;
Query OK, 0 rows affected (0.12 sec)
```

Ref: https://www.ibm.com/docs/en/ias?topic=features-database-user-management