DBMS

SQL QUERIES

NEELAM RAUF(110852)

OUTLINE

1. CHANGE BY USING ALTER
2. Day data_type
3. View in db
4. It's Types
5. How to use joins?
6. MySQL implementation

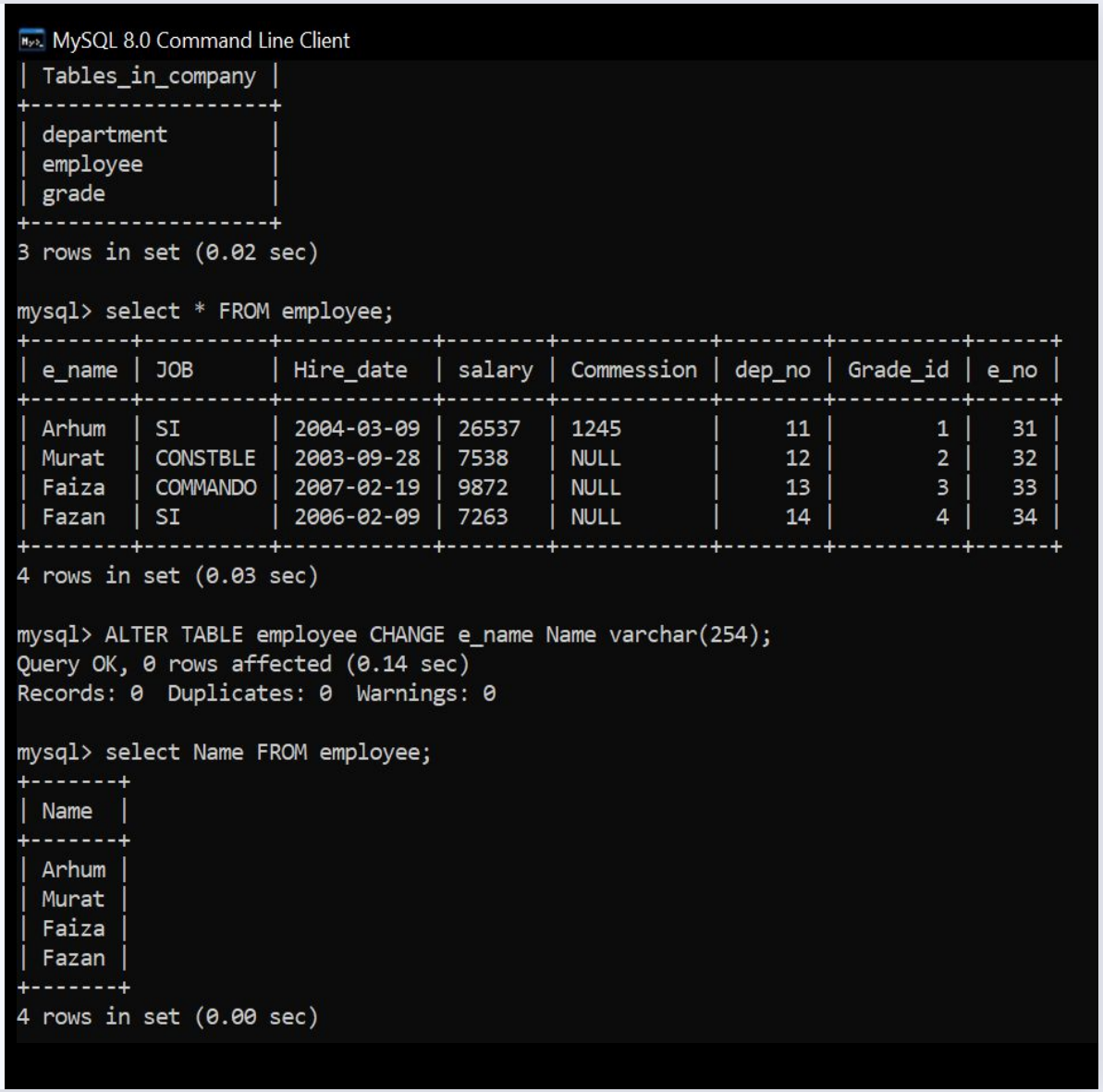
ASSIGNMENT

Essentially, it touches upon data definition (altering tables, views) and data manipulation (joins), along with basic user access control in MySQL.

CHANGING AN ATTRIBUTE (COLUMN):

Explanation:

To modify the definition of an existing column in a MySQL table, you use the **ALTER** TABLE statement. You can change its data type, constraints (like NOT NULL), and default value. There are two main clauses: MODIFY COLUMN (to change data type and constraints without renaming) and CHANGE COLUMN (to change the name, data type, and constraints).



ENTERING A DATATYPE OF "DAY":

Explanation:

MySQL doesn't have a direct "day" datatype as a standalone unit. You typically represent a day within a date using the DATE datatype or as a text string using VARCHAR. You could also use ENUM or SET for a predefined list of days.

```
1
mysql> Alter table employee ADD column DAY date;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> INSERT INTO employee VALUES("2023-12-20");
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> UPDATE employee SET DAY="2023-12-20" where e_no=32;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select DAY FROM employee where e_no=32;
+------------+
| DAY        |
+------------+
| 2023-12-20 |
+------------+
1 row in set (0.00 sec)
```

VIEW:

SIMPLE	COMPLEX	MATERIALIZED
Based on a single table	Based on multiple tables (joins)	
without aggregate functions or complex clauses	aggregate functions, GROUP BY	

View Command:

```
mysql> CREATE OR REPLACE VIEW employee_overview AS
    -> SELECT JOB ,Hire_date,salary,Commission,dep_no, Grade_id,e_no FROM employee;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from employee_overview
    -> ;
+---------+-----------+--------+-----------+--------+----------+------+
| JOB     | Hire_date | salary | Commission | dep_no | Grade_id | e_no |
+---------+-----------+--------+-----------+--------+----------+------+
| SI      | 2004-03-09 | 26537 | 1245      |     11 |        1 |   31 |
| CONSTBLE | 2003-09-28 | 7538 | NULL      |     12 |        2 |   32 |
| COMMANDO | 2007-02-19 | 9872 | NULL      |     13 |        3 |   33 |
| SI      | 2006-02-09 | 7263 | NULL      |     14 |        4 |   34 |
+---------+-----------+--------+-----------+--------+----------+------+
4 rows in set (0.01 sec)

mysql> CREATE USER 'arhum'@'localhost' IDENTIFIED BY '12345678';
Query OK, 0 rows affected (0.03 sec)
```

Explanation: You create a view using the CREATE VIEW statement followed by the view name and the SELECT query that defines the view.

Query on MySQL:

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

```
mysql> CREATE USER 'arhum'@'localhost' IDENTIFIED BY '12345678';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'arhum'@'localhost';
+---------------------------------------+
| Grants for arhum@localhost            |
+---------------------------------------+
| GRANT USAGE ON *.* TO `arhum`@`localhost` |
+---------------------------------------+
1 row in set (0.01 sec)
```

How can we provide write access to users on SQL?

Explanation:

You grant privileges to MySQL users using the GRANT statement. Write access typically involves **INSERT**, **UPDATE**, and **DELETE** privileges.

OTHER GRANT :

GRANT **INSERT**, **UPDATE**, **DELETE** ON database_name.table_name TO 'username'@'host';

FLUSH PRIVILEGES;

HOW CAN WE APPLY JOINS?

Explanation:

Joins are used to combine rows from two or more tables based on a related column. Different types of joins return different sets of rows.

JOINS are the crossproduct plus some conditions.

```
SELECT table1.column1, table2.column2
FROM table1
INNER JOIN table2 ON
table1.common_column =
table2.common_column;
```

JOINS:

INNER	LEFT	RIGHT
SELECT table1.column1, table2.column2 FROM table1	SELECT table1.column1, table2.column2 FROM table1	SELECT table1.column1, table2.column2 FROM table1
INNER JOIN table2 ON table1.common_column = table2.common_column	LEFT JOIN table2 ON table1.common_column = table2.common_column;	RIGHT JOIN table2 ON table1.common_column = table2.common_column;