Database Management Systems Training

June 12 - 16, 2023

One Central Hotel
Cebu City, Philippines

CREATE USER

- It is used to create a new user account in the database
- Syntax:

```
CREATE USER username IDENTIFIED BY password;
```

CREATE USER

Example

```
CREATE USER dbuser IDENTIFIED BY '1234';

CREATE USER dbuser@localhost IDENTIFIED BY '1234';
```

ALTER USER

- It is used to modify the properties or settings of an existing user account

- Syntax:

```
ALTER USER username [OPTIONS];
```

ALTER USER

Examples

```
--
-- Modify user password
--
ALTER USER 'dbuser' IDENTIFIED BY 'newpass'
;

--
-- Modify user password
-- with password expiration
--
ALTER USER 'dbuser' IDENTIFIED BY 'newpass'
  PASSWORD EXPIRE INTERVAL 90 DAY
;
```

ALTER USER

Examples

```
--
-- Lock user account
--
ALTER USER 'dbuser' ACCOUNT LOCK
;


--
-- Unlock user account
--
ALTER USER 'dbuser' ACCOUNT UNLOCK
;
```

DROP USER

- It is used to delete or remove a user account from a database
- Syntax:

```
DROP USER username;
```

DROP USER

Example

```
DROP USER dbuser;

DROP USER dbuser@localhost;
```

CREATE INDEX

- The CREATE INDEX statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly.

CREATE INDEX

- Syntax

```
CREATE INDEX index_name ON table_name (column1, column2, ...);

CREATE UNIQUE INDEX index_name ON table_name (column1, column2, ...);
```

CREATE INDEX

- Example

```
CREATE INDEX ix_name ON employee (lastname, firstname);

CREATE UNIQUE INDEX uix_idno ON employee (idno);
```

DROP INDEX

- The DROP INDEX statement is used to delete an index in a table

DROP INDEX

- Syntax

```
DROP INDEX index_name ON table_name;
```

DROP INDEX

- Example

```
DROP INDEX ix_name ON employee;

DROP INDEX uix_idno ON employee;
```

DML Commands

INSERT INTO

- It is used to insert new records into a table.
- It allows you to specify the values to be inserted into specific columns or provide values for all columns in the table.

INSERT INTO

- Syntax

```
INSERT INTO table_name (
    column1, column2, column3, ...
) VALUES (
    value1, value2, value3, ...
);
```

INSERT INTO

Examples

```
INSERT INTO employee (
    objid, idno, name, salary
) VALUES (
    'EMP01', '001', 'JUAN DELA CRUZ', 10000
);


INSERT INTO employee ( objid, idno, name, salary )
VALUES ( 'EMP01', '001', 'JUAN DELA CRUZ', 10000 ),
       ( 'EMP02', '002', 'JUAN ABUNDA', 11000 ),
       ( 'EMP02', '002', 'JOSE MANALOTO', 12000 )
;
```

UPDATE

- It is used to modify the existing records in a table.

- Syntax:

```
UPDATE table_name SET
    column1 = value1,
    column2 = value2,
    ...
WHERE
    condition
;
```

UPDATE

Examples

[MySQL]

```
UPDATE employee SET salary = 10000 WHERE objid = 'EMP001';


UPDATE order o, order_item oi
SET
    o.state = 'HOLD',
    o.remakrs = 'HOLD'
WHERE
    o.objid = oi.parentid AND
    oi.productid = 'PROD001'
;
```

UPDATE

Examples

```
[MSSQL]

UPDATE employee SET salary = 10000 WHERE objid = 'EMP001'
GO


UPDATE o SET
    o.state = 'HOLD',
    o.remakrs = 'HOLD'
FROM
    order o, order_item oi
WHERE
    o.objid = oi.parentid AND
    oi.productid = 'PROD001'
GO
```

DELETE

- It is used to delete existing records in a table.
- Syntax:

```
DELETE FROM table_name WHERE condition;
```

DELETE

Examples

```
DELETE FROM employee WHERE age < 18
;


DELETE FROM employee WHERE (salary = 0 OR salary IS NULL)
;


DELETE FROM employee
;
```

DCL Commands

GRANT

- It is used to grant specific privileges or permissions to users or roles in a database.
- It allows granting various levels of access to database objects, such as tables, views, procedures, or even the entire database.
- Syntax:

```
GRANT privilege(s) ON object TO user_or_role;
```

GRANT

Examples

```
GRANT ALL PRIVILEGES ON *.* TO admin_user;

GRANT SELECT, INSERT ON employee TO admin_user;

GRANT SELECT, INSERT ON etracs255.* TO admin_user;
```

REVOKE

- It is used to revoke or remove previously granted privileges or permissions from users or roles in a database.
- It allows you to withdraw specific access rights from database objects or completely remove a user's access to the database.
- Syntax:

```
REVOKE privilege(s) ON object FROM user_or_role;
```

REVOKE

Examples

```
REVOKE ALL PRIVILEGES ON *.* FROM admin_user;

REVOKE SELECT, INSERT ON employee FROM admin_user;

REVOKE SELECT, INSERT ON etracs255.* FROM admin_user;
```

TCL Commands

BEGIN

- It is used to mark the beginning of a transaction
- It establishes a transaction context, enabling the DBMS to keep track of the changes made within the transaction.
- It also sets a savepoint, which allows for partial rollbacks if needed.
- Syntax:

```
BEGIN;
```

BEGIN

Examples

```
BEGIN;
--
-- Data manipulation
-- statements go here
--
COMMIT;
```

COMMIT

- It is used to save changes made within a transaction.
- When executed, it confirms the transaction's changes and makes them permanent.
- Syntax:

```
COMMIT;
```

ROLLBACK

- It is used to undo or rollback the changes made within a transaction and restore the database to its previous state.
- Syntax:

```
ROLLBACK ;
```

ROLLBACK

Examples

```
BEGIN
;
--
-- Data manipulation
-- statements go here
--
-- Check for error or condition
-- that requires rollback
ROLLBACK
;
```

DQL Commands

SELECT

- It is used to retrieve data from one or more database tables.
- It allows you to specify the columns to retrieve, conditions for filtering data, sorting order, and more.

SELECT

Syntax

```
SELECT field1, field2, ...
FROM table_name
[WHERE Clause]
```

SELECT

Examples

```
-- display all fields
SELECT * FROM `order`
;

-- display selected fields
SELECT orderno, orderdate
FROM `order`
;
```

WHERE clause

- It is used to filter the rows returned by a query based on specific conditions.
- It allows you to selectively retrieve data that meets certain criteria from one or more database tables.

WHERE clause

Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
;
```

WHERE clause

Examples

```sql
-- filtering a specific value
SELECT column1, column2
FROM table_name
WHERE column1 = 'value'
;

-- using comparison operators
SELECT column1, column2
FROM table_name
WHERE column2 > 100
;
```

```sql
-- combining multiple conditions
SELECT column1, column2
FROM table_name
WHERE column1 = 'value'
    AND column2 > 100
;
```

LIMIT clause

- It is used to limit the number of rows returned by a query.
- It allows you to retrieve a specific number of rows or a subset of rows from a result set.

LIMIT clause

Syntax

```
SELECT column1, column2, ...
FROM table_name
LIMIT number_of_rows
;
```

LIMIT clause

Example

```
SELECT column1, column2
FROM table_name
LIMIT 10
;
```

```
SELECT column1, column2
FROM table_name
LIMIT 0, 10
;
```

ORDER BY clause

- It is used to sort the result set of a query based on one or more columns.
- It allows you to arrange the rows in either ascending (default) or descending order.

ORDER BY clause

Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC | DESC]
;
```

ORDER BY clause

Examples

```sql
-- sort by a single column
SELECT column1, column2
FROM table_name
ORDER BY column1
;


-- sort by multiple columns
SELECT column1, column2
FROM table_name
ORDER BY column1, column2 DESC
;
```

GROUP BY clause

- It is used to group rows in a result set based on one or more columns.
- It allows you to perform aggregate functions and generate summary information for each group.

GROUP BY clause

Syntax

```
SELECT
    column1, column2, ...,
    aggregate_function(column_name)
FROM table_name
GROUP BY column1, column2, ...
;
```

GROUP BY clause

Examples

```sql
-- group by a single column
SELECT department, SUM(salary)
FROM employees
GROUP BY department
;

-- group by multiple columns
SELECT department, gender, COUNT(*) as total
FROM employees
GROUP BY department, gender
;
```

HAVING clause

- It is used to filter the results of a GROUP BY query based on a condition applied to the aggregated values.
- It allows you to specify a condition for groups that should be included in the result set.

HAVING clause

Syntax

```
SELECT
    column1, column2, ...,
    aggregate_function(column_name)
FROM table_name
GROUP BY column1, column2, ...
HAVING condition
;
```

HAVING clause

Examples

```sql
-- Filter groups based on aggregated values
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary) > 5000
;

-- Combine aggregate functions with conditions
SELECT department, COUNT(*)
FROM employees
GROUP BY department
HAVING COUNT(*) > 10
    AND SUM(sales) > 10000
;
```

DISTINCT clause

- It is used in a SELECT statement to return only unique values in the result set.
- It eliminates duplicate rows from the query result.

DISTINCT clause

Syntax

```
SELECT DISTINCT
    column1, column2, ...
FROM table_name
;
```

DISTINCT clause

Examples

```
-- Retrieve distinct values from a single column
SELECT DISTINCT column1 FROM table_name
;

-- Retrieve distinct values from multiple columns
SELECT DISTINCT column1, column2 FROM table_name
;
```

The Complete Syntax of a SELECT SQL Statement

```
select
    [ distinct ]
    column(s)

from
    table_name(s)

[ where condition(s) ]

[ group by column(s) having condition(s) ]

[ order by column(s) ]

[ limit number_of_rows ]
```

Working on SQL VIEW

CREATE VIEW

- A view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

CREATE VIEW

Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition
;
```

CREATE VIEW

Examples

```
CREATE VIEW vw_employee AS
SELECT objid, idno, name
FROM employee
;

CREATE VIEW vw_employee AS
SELECT objid, idno, name
FROM employee
WHERE age >= 18
;
```

DROP VIEW

- To delete a view in a database
- Syntax

```
DROP VIEW view_name;
```

DROP VIEW

Examples

```
DROP VIEW vw_employee
;


DROP VIEW IF EXISTS vw_employee
;
```