07/11/22 Experiment 1 : DDL commands

Aim : Performing DDL commands like CREATE, DROP, ALTER, TRUNCATE, RENAME based on conditions

Theoretical Background

DDL is an abbreviation of Data Definition Language. It is used to create and modify the schema of the database and its objects. Following are the five DDL commands in SQL

1. CREATE command
2. DROP command
3. ALTER command
4. TRUNCATE command
5. RENAME command

CREATE command

It is a DDL command used to create database, tables, triggers and other database objects

Syntax : CREATE DATABASEOBJECT OBJECTNAME;

Example: CREATE TABLE books;

To add columns

Syntax : CREATE TABLE TABLENAME (columnname1 datatype, columnname2 datatype ... columnnamen datatype);

DROP Command

It is used to delete / remove the database objects from the SQL database. We can easily remove the entire table, view or index from the database using this DDL command.

Syntax: DROP databaseObject Objectname;

example: DROP TABLE books;

ALTER command

It is a DDL command which changes or modifies the existing structure of the database, and it also changes the schema of database objects. We can also add or drop constraints of the table using ALTER command.

Syntax: ALTER TABLE TABLENAME

1. ADD columnname datatype - Adds a new column
2. MODIFY columnname datatype - Modifies existing column
3. DROP columnname - Deletes/Removes column
4. ADD PRIMARY KEY (columnname)
5. ADD FOREIGN KEY(columnname)
6. ADD CONSTRAINT - Adds constraint to column

Example: ALTER TABLE Books

Add columnbooks varchar(20);

Adds a column named columnbooks of varchar datatype.

TRUNCATE command

It is another DDL command which delete or removes all the records from the table. This command also removes the space allocated for storing the table records.

Syntax: TRUNCATE TABLE TABLENAME;

RENAME command

It is a DDL command used to change the name of a table or a column inside the table.

Syntax: RENAME TABLE oldname TO newname;

Example: RENAME TABLE Books TO Bookdetails;

To change column name,

① ALTER TABLE Bookdetails RENAME COLUMN BID TO ID;

② ALTER TABLE Bookdetails CHANGE COLUMN oldname newname datatype;

DUAL TABLE In SQL

In Oracle, clause FROM is not exceptional. If we don't write the FROM clause, we'll get an error.

for example, SELECT SysDate;

output: FROM keyword not found where expected

DUAL: It is a table that is automatically created by Oracle Database along with the data dictionary. DUAL is the schema of the user SYS but is accessible by the name DUAL to all users. It has one column, DUMMY, defined to be VARCHAR(1) and contains one row with a value X.

Example: SELECT * FROM DUAL;

output: X

Selecting from DUAL is useful for computing a constant expression with the SELECT statement.

Example: SELECT 'Books' AS NAME FROM DUAL;

Output: Books

Example: SELECT 2+2 FROM DUAL;

output: 2+2 = 4

Several databases including MS SQL, mySQL etc allows the omitting of FROM clause. This exception is the reason there is no dummy table like DUAL in other databases.

Conclusion

DDL commands executed and output verified for various

21/11/22 Experiment 2: DML Commands 6

Aim: Performing DML commands like Insertion, Deletion, Modifying, Altering and Updating records based on conditions

Theoritical Background

DML is an abbreviation of data manipulation language. It is used in structured query language to change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands. The 4 main DML commands in SQL are

1. SELECT command
2. INSERT command
3. UPDATE command
4. DELETE command

SELECT Command

It shows the records of the specified table. It also shows the particular record of a particular column by using the WHERE clause.

Syntax: SELECT COLUMNNAME FROM TABLENAME : selects column
SELECT * FROM TABLENAME : Selects all records
SELECT DISTINCT COLUMNNAME FROM Tablename : Selects unique values
SELECT * FROM Tablename WHERE condition

Example: SELECT * FROM Books.

We can also use it to display count of certain elements / records.

Syntax: SELECT count(*) from tablename

Other clauses that can be used with SELECT are:

i) GROUP By:

Syntax: Select * from tablename groupby columnname

HAVING CLAUSE

Syntax: Select * from tablename group by columnname having condition

ii) ORDER BY : Arranges in ascending or descending order

Syntax : Select * from tablename ~~where~~ order by columnname desc;

INSERT Command

It allows users to input data in database tables.

Syntax : INSERT INTO TABLENAME VALUES (value1, value2 ... valuen);

UPDATE Command

This command allows users to update or modify the existing data in database tables

Syntax: UPDATE tablename SET Columnname = 'value' where condition;

Example: UPDATE books SET bookname = 'Start' where ID = 001;

DELETE command

It allows user to remove single or multiple existing records from the database tables. This command does not delete the stored data pumanently from the database. We can use the where clause with DELETE command to select specific rows from the table

Syntax : DELETE FROM Tablename WHERE condition;

Example : DELETE FROM Books WHERE BID = 002;

Comparison operators

operators that can be used with SQL queries are: =, >=, <=, <, >, <> (not equal) and the logical operators AND, OR, NOT.

Other commands

- IS NULL - checks if column value is null or not
- BETWEEN, AND - gets a range of values.

Relational Algebra

It takes relation as input and generates relation as output.

① Projection (Π) - Used to project required column data from a relation

Syntax: Π colname (Tablename)

② Selection (σ) - Used to select required tuples of relation

Syntax: σ condition (Tablename)

③ Rename (ρ) : Renames table or column

Syntax: Newname ← Tablename / σ / Π

④ Union (∪) : Same as set theory however both relations must have same set of attributes. Syntax: Table1 ∪ Table2

⑤ Intersection (∩) : Syntax : Table1 ∩ Table2.

⑥ Minus (−) / Set Difference : Same as set theory

⑦ Cross product (X) : Results would each attribute of A followed by B

⑧ Natural join (⋈) : Binary operators that joins two relations.

Syntax: Table1 ⋈condition Table2.

Aggregate functions

① Maximum — $\mathcal{F}_{max}$ colname (Tablename)

② minimum — $\mathcal{F}_{min}$ Colname (Tablename)

③ Average — $\mathcal{F}_{Avg}$ Colname (Tablename)

④ Count — $\mathcal{F}_{count}$ Colname (Tablename)

Conclusion

Performed various DML commands and verified the outputs of the queries.

29/11/22 Experiment 3 : Views and Built in functions 12

Aim : Performing commands using views and in built functions

Theoritical Background

A view is a virtual table in the database defined by a query. It does not exist in the database as a stored set of data values.

Syntax : CREATE VIEW viewname columnname1, columnname2 ....
AS query

Types of Views

① Horizontal Views - Restricts a user's access to selected rows of table

for example, CREATE VIEW view_cust AS
Select * from customerdetails where CID = 103 ;

② Vertical Views - Restricts a user's access to select columns of table

For example, CREATE VIEW Cust AS
Select ~~Cust~~ID from customer_details ;

DROP VIEW : Drops a view

Syntax : DROP VIEW viewname ;

Joined Views : Joined views are used to simplify multi table queries.

For example, CREATE VIEW cust AS
Select custdetails, custname, fixeddepoit
from customerdetails, Custfixeddeposit
where customerdetails.CustID = Custfixeddeposit.CustID ;

Advantages of Views

- Security
- Query simplicity and Structural simplicity
- Data Integrity

Disadvantages of views

- Performance
- Update restrictions

View updates

A view can be updated if the following conditions are met:

→ DISTINCT clause must not be specified

→ FROM clause must specify only one updateable table

→ The SELECT list cannot contain expressions, calculated columns or column functions

→ The WHERE clause must not include a sub query

→ The SELECT list must include all the columns specified with the NOT NULL constraint

Oracle Built-In functions

① Single row functions / Scalar functions returns a value for each row processed in query

② Group functions group the rows of data based on values returned by query

Types of single row functions

1) Numeric function : They accept numeric data and return numeric values.

2) Character / Text functions : Accept character input, return both numeric and character values

3) Date functions : Takes and returns values of date datatype except for the months_between function

4) Conversion functions : converts a value of form to another form.

Numeric functions

Function name	Return Value
ABS (x)	Absolute value of the number 'x'
CEIL (x)	Integer value that is greater or equal to the number 'x'
FLOOR (x)	Integer value that is lower or equal to the number 'x'
TRUNC (x, y)	Truncates value of x upto y decimals
ROUND (x, y)	Rounded off value of number 'x' upto 'y' decimal places

Character or Text functions

Function name	Return Value
LOWER (string value)	All letters in string are converted to lowercase
UPPER (string)	All letters in string are converted to uppercase
INITCAP (string)	All letters in string are converted to mixed case
LTRIM (string, text)	'text' text is removed from left of string
RTRIM (string, text)	'text' text is removed from right of string
TRIM (text FROM value)	'text' can be one character long and will be removed
SUBSTR (value, m, n)	Returns n characters from value starting from m position
LENGTH (value)	Number of characters in string
LPAD (string, n, value)	Returns string left padded with value.
RPAD (string, n, value)	Return string right padded with value.

Date functions

ADD_MONTHS (date, n)

MONTHS_BETWEEN (x1, x2)

ROUND (x, date_format)

TRUNC (x, dateformat)

NEXT_DAY (x, week_day)

LAST_DAY (x)

SYSDATE

NEW_TIME (x, zone1, zone2)

conversion functions

TO_CHAR (x [,y]) - converts numeric and date values to string value

TO_DATE (x [, dateformat]) - converts to a date value

NVL (x, y) - Null value 'x' is replaced by 'y' of same datatype

DECODE (a, b, c, d, e, default) - checks value of 'a'. if a=b then returns c else if a=d then returns e else returns default.

conclusion

Performed and executed commands using views and in built functions. ✓

Q3 13/12/22

29/11/22

Experiment 4 : PL/SQL Programs

Aim : Implementing the given programs using PL/SQL blocks.

Theoritical background

A subprogram is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. PL/SQL provides two kinds of subprogram

- **Functions** : These programs return a single value
- **Procedures** : These are used to perform an action

Parts of PL/SQL

1) **Declarative Part** : This contains declarations of items local to the subprogram and cease to exist when the subprogram completes execution
2) **Executable Part** : It contains statements that perform designated actions
3) **Exception handling** : It contains the code that handles run-time errors.

Parameter modes in PL/SQL

1) **IN** : It is a read only parameter. It is the default mode of parameter passing. Parameters are passed by reference
2) **OUT** : The actual parameter must be variable and it is passed by value.
3) **IN OUT** : It passes an initial value to the subprogram and returns an updated value to the caller. Actual parameter is passed by value

Example

```
create or replace procedure findmin (x in number, y in number, z out
number) is
begin
    if  x < y then
        z := x;
```

Aim:

To execute the given subprograms using Procedures in PL/SQL

Theoritical Background

Procedures

Syntax:

```
CREATE [OR REPLACE] PROCEDURE procedurename
[(parametername [IN|OUT|INOUT] type [,...])]
{IS|AS}
BEGIN
    < procedure body >
END Procedurename;
```

where: • procedurename specifies the name of the procedure
• [OR REPLACE] option allows modification of an existing procedure
• procedure_body contains executable part
• The AS keyword is used for creating a standalone procedure
• The optional parameter list contains name, mode and types of parameter

for example,

```
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
    dbms_output.put_line ('Hello');
END;
/
EXECUTE greetings;
```

Procedures can be deleted using drop procedure statement.

Syntax: DROP PROCEDURENAME;

for example DROP greetings;

6/12/22

Experiment 6: Aggregate functions, Nested Queries and Join Queries

Aim

Implementation of various aggregate functions, set operators, nested queries and join queries in SQL.

Theoretical background

Aggregate functions

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning. The various aggregate functions are

1) count() : Select count (colname) from table;
2) sum() : Select sum (colname) from table;
3) avg() : Select avg (colname) from table;
4) min() : Select min (colname) from table;
5) max() : Select max (colname) from table;

Nested query

In nested queries, a query is written inside a query. The result of inner query is used in execution of outer query. There are mainly two types of nested queries:

- Independent nested queries: Query execution starts from innermost query to outermost queries. Various operations are IN, NOT, IN, ANY, ALL etc are used in writing independent nested queries.
- Correlated nested queries: In correlated nested queries, the output of inner query depends on the row which is being currently executed in outer query.

Join Queries

A join clause is used to combine rows from two or more tables based on a related column between them. The different types of SQL joins are

- (INNER) join : Returns records that have matching values in both tables.
- LEFT (OUTER) join : Returns all records from the left table and the matched records from the right table.
- RIGHT (OUTER) join : Returns all records from the right table, and the matched records from the left table.
- FULL (OUTER) join : Returns all records when there is a match in either left or right table.

Experiment 7 : Functions

Aim:
To execute the given programs using functions in PL/SQL

Theoritical background

Functions

Syntax

```
create [or replace] function function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
Return return_datatype
{IS | AS}
begin
    < function_body >
end [function_name];
```

where
- function_name specifies the name of the function
- [OR REPLACE] option allows the modification of an existing function
- The optional parameter list contains name, mode and type of the parameter.
- The function must contain a return statement
- The return clause specifies the data type you are going to return from the function
- function_body contains the executable part
- The AS keyword is used instead of the IS keyword for creating a standalone function.

For example,

```
create or replace function totalcust
return number IS
    total number(2) := 0;
begin
```

Experiment 8 : cursors

Aim

To execute the given functions using cursors

Theoretical Background

A cursor is a handle or pointer to the context area. Through the cursor a PL/SQL program can control the context area and what happens to it as the statement is processed. Important features of the cursor are :

1) cursors allow you to fetch and process rows returned by SELECT statement, one row at a time.
2) A cursor is named so that it can be referenced.

There are two types of cursors

1) An implicit cursor is automatically declared by Oracle every time an SQL statement is executed. The user will not be aware of this happening and will not be able to control or process the information in an implicit cursor.
2) An explicit cursor is defined by the program for any query that returns more than one row of data. That means the programmer has declared the cursor within the PL/SQL block.

Declaring a cursor

Declaring a cursor defines the name of the cursor and associates it with a SELECT statement. The first step is to declare the cursor with the following syntax

```
CURSOR cursorname IS select statement ;
```

cursor names follow the same rules as PL/SQL identifiers. Because the name of the cursor is a PL/SQL identifier it must be declared before it is referenced.

Record types

A record is a composite data structure which means that it is composed of more than one element. Records are very much like a row of a database table but each element of the record does not stand on its own. PL/SQL supports three kinds of records

1. Table based - drawn from the list of columns in a table
2. cursor based - record whose structure matches a predefined cursor
3. Programmer defined

Syntax: <record_name> <tablename or cursorname> %ROWTYPE;

Explicit cursor attributes

Cursor attribute	Syntax	Explanation
%NOTFOUND	cursorname%NOTFOUND	A boolean attribute that returns TRUE if previous fetch did not return a row and false if it did.
%FOUND	cursorname%FOUND	A boolean attribute that returns TRUE if previous fetch returned a row and false if it did not.
%ROWCOUNT	cursorname%~~NOTFOUND~~ ROWCOUNT	# of records fetched from a cursor at that point of time
%ISOPEN	cursorname%ISOPEN	A boolean attribute that returns TRUE if cursor is open, FALSE if it is not.

Cursors with parameters

A cursor can be declared with parameters. This enables a cursor to generate specific results which is on one hand

24/01/23

Experiment 9. TCL Commands

Aim

To execute and implement various TCL commands

Theoritical Background

TCL is acronym for Transaction control language. They are used for managing and controlling the transactions in a database to maintain consistency.

A set of SQL statements executed on the information and data stored in DBMS is known as transaction. Any transaction when made happens temporarily in the DB. TCL commands are used to make these alterations pumanent.

There are 3 types of TCL commands

① COMMIT

The command helps user to save a transaction into the DB pumanently. DML commands like DELETE, UPDATE or INSERT alterations are not pumanent, meaning they roll back if not stored pumanently. Commit marks these changes as pumanent.

② ROLLBACK

This command functions to restore the DB to the very committed state. Alterations made that are unrequired can use ROLLBACK command to rollback the data to the last committed state.

Syntax: ROLLBACK TO savepoint_name;

③ SAVEPOINT

This command helps to save a transaction temporarily so that user can rollback to that point when required.

Syntax: SAVEPOINT savepoint_name;

Advantages of TCL

- It is used in SQL which is very interactive, portable and works in multiple programs
- Users can easily, access, edit and save changes to database.
- The coding required is not very complex.
- The commands are easy to remember.

Disadvantages of TCL

- The ROLLBACK command is of no use in cases where user commits using the COMMIT command.
- One has to create a savepoint in order to go back to it or else a chunk of current changes will go in vain.
- Identifying save points may be confusing to some users.

Result

Output is obtained and verified

24/01/23 Experiment 10: Triggers

Aim

To execute various PL/SQL programs using Triggers

Theoritical Background

A trigger is a special kind of stored procedure procedure that executes in response to certain action on the table like insertion, deletion or updating of data.

ECA Model

1) Event
   - The rules are triggered by event

2) Condition
   - The condition that determines whether the rule action should be executed.
   - If a condition is specified it is first evaluated and only if it evaluates to true will the rule action be executed.

3) Action
   - Usually a sequence of SQL statements.
   - It could also be a database transaction or an external program that will be automatically executed.

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
{OF col_name]
ON table_name
```

[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
    Declaration - statements
BEGIN
    Executable - statements
EXCEPTION
    Exception - handling - statements
END;

Output is contained and executed