Lab3

Objective

To understand and implement Data Query Language (DQL) concepts, relational algebra operations, and Query By Example (QBE) for effective database querying and management.

Theory

Data Query Language (DQL)

DQL is a key subset of SQL (Structured Query Language) designed for retrieving data from a database without modifying it. It is essential for data analysis and retrieval operations.

Key Features of DQL

1. Data Retrieval: Allows fetching specific data from tables based on user-defined conditions.
2. Non-Destructive Operations: Performs read-only operations, ensuring the original data remains unchanged.
3. Powerful Query Capabilities: Enables filtering (`WHERE`), aggregation (`COUNT`, `AVG`), grouping (`GROUP BY`), and sorting (`ORDER BY`) for customized queries.
4. Core Command - SELECT: Used for retrieving specific rows and columns from one or more tables.

Common DQL Commands

1. SELECT: Retrieves specific columns from a table.
    - Syntax: `SELECT column_names FROM table_name WHERE condition;`
2. ORDER BY: Sorts the result set in ascending (`ASC`) or descending (`DESC`) order.
    - Syntax: `SELECT * FROM table_name ORDER BY column_name [ASC|DESC];`
3. WHERE Clause: Filters rows based on conditions.
    - Syntax: `SELECT * FROM table_name WHERE condition;`
4. LIKE: Performs pattern matching in string columns.
    - Syntax: `SELECT * FROM table_name WHERE column_name LIKE 'pattern';`
5. DISTINCT: Removes duplicate rows from the result.
    - Syntax: `SELECT DISTINCT column_name FROM table_name;`

Relational Algebra

Relational algebra provides a theoretical basis for SQL and defines operations to manipulate and retrieve data from relational databases.

Key Operations

1. Selection (σ): Filters rows based on a condition.
    - Example: σ_{column = value}(relation)
2. Projection (Π): Selects specific columns from a relation.
    - Example: Π_{column1, column2}(relation)
3. Renaming (ρ): Renames attributes or relations for clarity.
    - Example: ρ_{new_name/old_name}(relation)

Query By Example (QBE)

QBE is a high-level, visual approach to database querying where users specify desired results using a grid-like interface rather than writing SQL code. It simplifies database interactions for non-technical users.

Key Features of QBE

1. Visual Query Design: Users create queries by filling in conditions in the query grid.
2. Simplified Syntax: Reduces the need to write complex SQL statements manually.
3. Graphical Interface: Ideal for beginners and non-technical users due to its intuitive design.
4. Underlying SQL Translation: The database management system (DBMS) converts QBE inputs into equivalent SQL statements.

Components of QBE

1. Tables: Represent database tables displayed visually.
2. Query Grid: Contains columns for defining criteria like selection, filtering, and sorting.
3. Criteria Rows: Users enter conditions for data retrieval (e.g., specific values, patterns).
4. Action Rows: Define operations like sorting or displaying distinct values.

Example: QBE for Filtering and Sorting

- To retrieve all employees in the "HR" department, sorted by `LastName`:
    - Field: `Department`, `LastName`
    - Table: `Employees`
    - Criteria: `"HR"`
    - Sort: Ascending

Relationship Algebras

1. No Relationship Algebra
2. ρ(STDINFO , Studnent)
   ρ(first_name/f_name)(STDINFO)
3. NewTable ← Π_{s_id, first_name}(STDINFO)
4. Π_{s_id, first_name}(σ_{municipality='Pokhara'}(STDINFO))
5. Π_{s_id, first_name}(σ_{municipality != 'Pokhara'}(STDINFO))
6. Π_{s_id, first_name}(σ_{municipality='Pokhara' ∨ municipality='Kathmandu'}(STDINFO))
7. Π_{s_id, first_name}(σ_{municipality='Pokhara' ∨ s_id=7}(STDINFO))
8. Π_{s_id, first_name}(σ_{municipality ∈ {'Pokhara', 'Kathmandu', 'Lalitpur'}}(STDINFO))
9. Π_{s_id, first_name}(σ_{first_name LIKE 'a%'}(STDINFO))
10. Π_{s_id, first_name}(σ_{first_name LIKE '%v'}(STDINFO))
11. Π_{s_id, first_name}(σ_{first_name LIKE 'a%a'}(STDINFO))
12. Π_{s_id, first_name}(σ_{m_name IS NULL}(STDINFO))
13. Π_{wardno}(STDINFO)
14. δ(Π_{wardno}(STDINFO))
15. τ_{wardno ASC}(STDINFO)
16. τ_{wardno ASC, municipality ASC}(STDINFO)
17. Π_{s_id, b_name, b_price}(σ_{1000 ≤ b_price ≤ 1700}(BOOK))
18. Π_{s_id, b_name, b_price}(σ_{b_price > 1700}(BOOK))
19. Π_{s_id, b_name, b_price}(σ_{b_price < 1500}(BOOK))

QBE

1. Create table and input data

No QBE

2. Change the name of the table and column

No QBE

3. Copying Data to another Table

No QBE

4. If Municipality = Pokhara

STDINFO

```
municipality    | wardno | s_id | name    | m_name | l_name

'Pokhara'       |        | P._x | P._y    |        |
```

5. If Municipality not = Pokhara

STDINFO

```
municipality     | wardno | s_id | name    | m_name | l_name

!= 'Pokhara'     |        | P._x | P._y    |        |
```

6. If Municipality = Pokhara or Kathmandu

STDINFO

```
municipality                | wardno | s_id | name    | m_name |
l_name

'Pokhara' OR 'Kathmandu'      |       | P._x | P._y    |       |
```

7. If Municipality = Pokhara or s_id = 7

STDINFO

```
municipality     | wardno | s_id | name    | m_name | l_name

'Pokhara'        |        | P._x | P._y    |        |

                 |        | 7    | P._z    |        |
```

8. If Municipality is in (List)

STDINFO

```
municipality                              | wardno | s_id | name
| m_name | l_name

'Pokhara'  OR  'Kathmandu'  OR  'Bhaktapur'  OR  'Biratnagar'  |
| P._x | P._y    |        |
```

9. If first_name starts with “a”

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| | P._x | 'a%' | |

10. If first_name ends with “v”

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| | P._x | '%v' | |

11. If first_name starts with and ends with “a”

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| | P._x | 'a%a' | |

12. If m_name is null

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| | P._x | P._y | NULL |

13. Show all ward numbers

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| P._x | | | |

14. Show distinct Ward numbers

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| P._x | | | |

15. Order data by wardnumber

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| P._x | | | |

16. Order data by wardnumber and municipality

STDINFO

municipality | wardno | s_id | name | m_name | l_name

| P._x | | | |

17. If price between 1000 and 1700

BOOK

b_id | b_name | b_author | b_price | s_id

P._x | P._y | | _n | P._z

Conditions: 1000 <= _n <= 1700

18. If price more than 1700

BOOK

b_id | b_name | b_author | b_price | s_id

P._x | P._y | | _n | P._z

Conditions: _n > 1700

19. If price less than 1500

BOOK

b_id	b_name	b_author	b_price	s_id
P._x	P._y		_n	P._z

Conditions: _n < 1500

Discussion:

This lab was about learning SQL and relational algebra to work with databases. We practiced creating tables, inserting data, and writing queries to retrieve specific results without modifying the data. We also renamed tables and columns to make them more understandable. By applying relational algebra concepts like selection (filtering rows) and projection (selecting columns), we understood the basic logic behind SQL commands. Additionally, we sorted and filtered data using conditions such as AND, OR, and NOT, demonstrating how SQL can be used to manage data efficiently.

Conclusion:

We hereby conclude our lab on SQL and relational algebra.