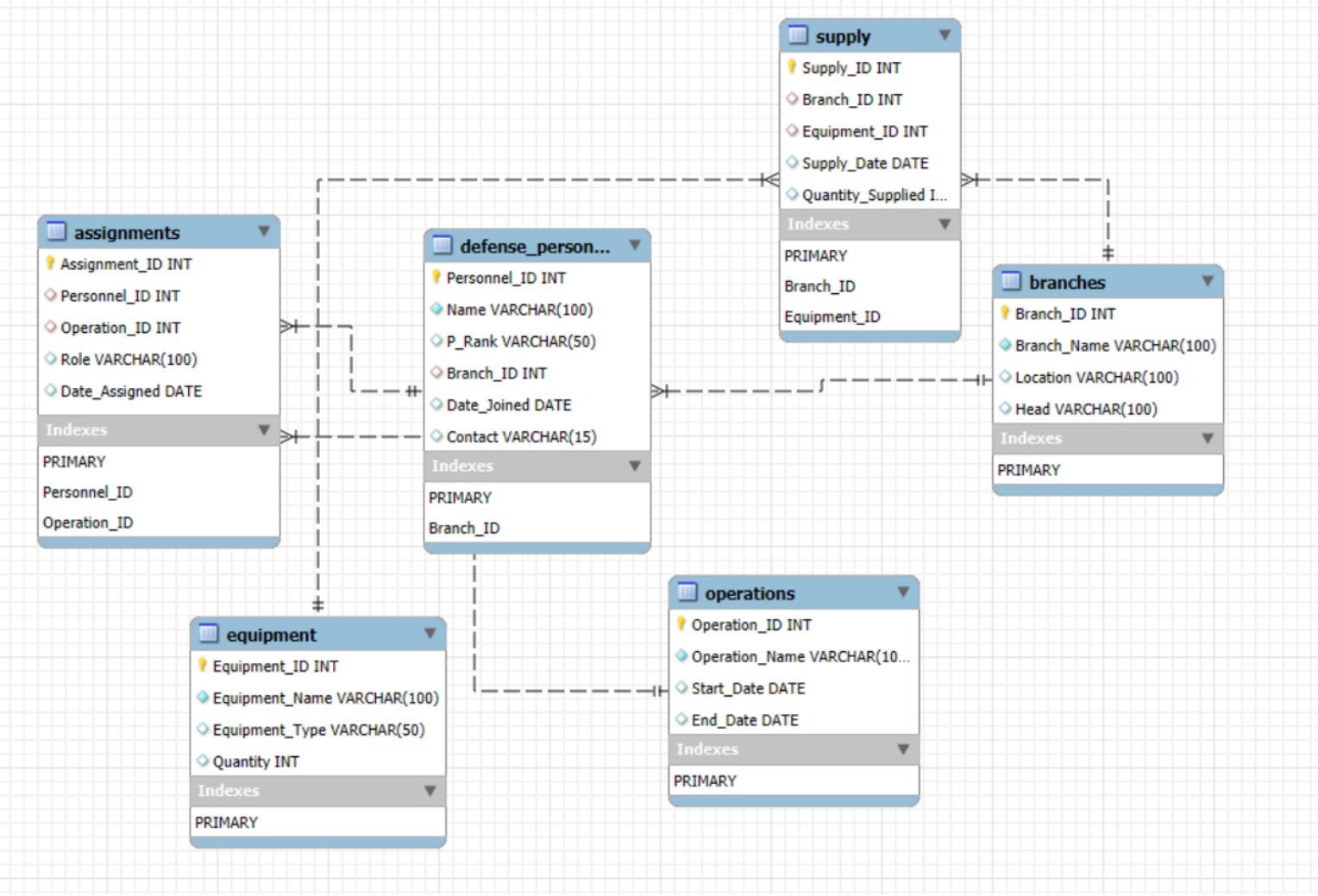
Project on Indian Defense Services (DBMS CIA3)

The project models the Indian defense services, tracking personnel, branches, operations, and the equipment supplied to various branches. It supports the management of operations, equipment inventory, and personnel assignments, ensuring efficiency and accuracy in defense logistics.

ER Diagram:

- **Entities**:
  - **Defense Personnel**: Attributes: Personnel_ID (PK), Name, Rank, Branch, Date_Joined, Contact
  - **Branches**: Attributes: Branch_ID (PK), Branch_Name, Location, Head
  - **Operations**: Attributes: Operation_ID (PK), Operation_Name, Start_Date, End_Date
  - **Equipment**: Attributes: Equipment_ID (PK), Equipment_Name, Equipment_Type, Quantity
  - **Assignments**: Attributes: Assignment_ID (PK), Personnel_ID (FK), Operation_ID (FK), Role, Date_Assigned
  - **Supply**: Attributes: Supply_ID (PK), Branch_ID (FK), Equipment_ID (FK), Supply_Date, Quantity_Supplied

- **Relationships:**
    - Personnel is **assigned to** Operations.
    - Branches **use** Equipment.
    - Personnel **belong to** a Branch.
    - Branches **are assigned** Equipment through Supply.



1. **DDL Commands (Data Definition Language)**
   Defines and modifies database structures like tables, indexes, views. Common commands include `CREATE`, `ALTER`, `DROP`, and `RENAME`.

Input:

```
-- Create a new table for Defense Equipment
CREATE TABLE Equipment (
    Equipment_ID INT PRIMARY KEY,
    Equipment_Name VARCHAR(100) NOT NULL,
    Equipment_Type VARCHAR(50),
    Quantity INT
);

-- Alter the Equipment table to add a new column
ALTER TABLE Equipment ADD Equipment_Description VARCHAR(200);

-- Rename the Equipment table to Defense_Equipment
RENAME TABLE Equipment TO Defense_Equipment;

-- Drop the table if no longer needed
DROP TABLE Defense_Equipment;
```

Output:

```
Branch_ID   | Branch_Name   | Location       | Head
-------------------------------------------------------
1           | Army          | New Delhi      | General Vipin Rawat
```

2. **DML Commands (Data Manipulation Language)**
   Used for managing data within tables. Commands include

INSERT, UPDATE, DELETE, and SELECT to manipulate and retrieve data.

Input:

```
-- Insert data into the Defense Personnel table

INSERT INTO Defense_Personnel (Personnel_ID, Name, Rank, Branch_ID,
Date_Joined, Contact)

VALUES (4, 'Ravi Sharma', 'Lieutenant', 1, '2010-05-15', '9876543240');


-- Update the Rank of a Personnel

UPDATE Defense_Personnel SET Rank = 'Captain' WHERE Personnel_ID = 4;


-- Delete a record from Defense Personnel

DELETE FROM Defense_Personnel WHERE Personnel_ID = 4;
```

Output:

```
Personnel_ID | Name        | Rank      | Branch_ID | Date_Joined  | Contact

-------------------------------------------------------------------------

4            | Ravi Sharma | Lieutenant| 1         | 2010-05-15   | 9876543240


Personnel_ID | Name        | Rank    | Branch_ID | Date_Joined  | Contact

-----------------------------------------------------------------------

4            | Ravi Sharma | Captain | 1         | 2010-05-15   | 9876543240


(No matching record found)
```

3. **TCL Commands (Transaction Control Language)**
   Controls transactions in SQL, ensuring the integrity of operations. Commands like COMMIT, ROLLBACK, and SAVEPOINT are used to manage transaction boundaries.

Input:

```
-- Start a transaction

START TRANSACTION;


-- Insert a record into Defense Personnel

INSERT INTO Defense_Personnel (Personnel_ID, Name, Rank, Branch_ID,
Date_Joined, Contact)

VALUES (5, 'Aman Joshi', 'Major', 2, '2012-08-20', '9876543250');


-- Save the transaction point

SAVEPOINT save1;


-- Insert another record into Defense Personnel

INSERT INTO Defense_Personnel (Personnel_ID, Name, Rank, Branch_ID,
Date_Joined, Contact)

VALUES (6, 'Vikram Singh', 'Major', 3, '2014-03-10', '9876543260');


-- Rollback to the previous savepoint

ROLLBACK TO save1;


-- Commit the transaction

COMMIT;
```

Output:

```
Personnel_ID | Name         | Rank   | Branch_ID | Date_Joined  | Contact

-------------------------------------------------------------------------

5            | Aman Joshi   | Major  | 2         | 2012-08-20   | 9876543250
```

4. VDL Commands (View Definition Language)

Used for creating and managing views in a database, providing a way to encapsulate complex queries or present

data in a specific format. Includes commands like `CREATE VIEW`, `DROP VIEW`, and `ALTER VIEW`.

Input:

```
-- Create a view of personnel in the Army branch
CREATE VIEW ArmyPersonnel AS
SELECT Name, Rank, Date_Joined
FROM Defense_Personnel
WHERE Branch_ID = 1;


-- View the Army personnel
SELECT * FROM ArmyPersonnel;


-- Drop the view
DROP VIEW ArmyPersonnel;
```

Output:

```
Name          | Rank          | Date_Joined
---------------------------------------
Amit Singh    | Colonel       | 2005-03-14
```

5. **Types of Joins**
   Combines rows from two or more tables based on a related column. Includes:
   - **INNER JOIN: Returns rows with matching values in both tables.**
   - **LEFT JOIN: Returns all rows from the left table and matched rows from the right table.**
   - **RIGHT JOIN: Returns all rows from the right table and matched rows from the left table.**

- **FULL OUTER JOIN**: Returns all rows when there is a match in one of the tables.

Input:

```
-- Inner Join between Defense Personnel and Branches
SELECT dp.Name, dp.Rank, b.Branch_Name
FROM Defense_Personnel dp
INNER JOIN Branches b ON dp.Branch_ID = b.Branch_ID;

-- Left Join to show all personnel and their branches
SELECT dp.Name, dp.Rank, b.Branch_Name
FROM Defense_Personnel dp
LEFT JOIN Branches b ON dp.Branch_ID = b.Branch_ID;

-- Full Outer Join (if supported, otherwise simulate using UNION)
SELECT dp.Name, dp.Rank, b.Branch_Name
FROM Defense_Personnel dp
FULL OUTER JOIN Branches b ON dp.Branch_ID = b.Branch_ID;
```

Output:

```
Name          | Rank         | Branch_Name
-----------------------------------------
Amit Singh    | Colonel      | Army

Name          | Rank         | Branch_Name
-----------------------------------------
Amit Singh    | Colonel      | Army
Suresh Mehra  | Commander    | Navy
Rajesh Kumar  | Wing Commander | Air Force
```

6. **Where Clause**
   Filters records that meet specific conditions. It is used in `SELECT`, `UPDATE`, and `DELETE` queries to restrict the rows affected by the command.

Input:

```
SELECT * FROM Defense_Personnel
WHERE Rank = 'Colonel';
```

Output:

```
Personnel_ID | Name         | Rank    | Branch_ID | Date_Joined  | Contact
-------------------------------------------------------------------------
1            | Amit Singh   | Colonel | 1         | 2005-03-14   | 9876543210
```

7. **Group By and Having Clauses**
   - **Group By**: Groups rows that have the same values into summary rows, often used with aggregate functions (`COUNT`, `SUM`).
   - **Having**: Similar to `WHERE`, but used to filter aggregated data after `GROUP BY`.

Input:

```
-- Group personnel by branch and count
SELECT Branch_ID, COUNT(*) AS Personnel_Count
FROM Defense_Personnel
GROUP BY Branch_ID
HAVING COUNT(*) > 1;
```

Output:

```
Branch_ID   | Personnel_Count
-----------------------------
1           | 2
2           | 1
```

8. Order By Clause

Specifies the order in which records are returned in a query, either in ascending (ASC) or descending (DESC) order.

Input:

```
SELECT * FROM Defense_Personnel
ORDER BY Date_Joined DESC;
```

Output:

```
Personnel_ID | Name         | Rank         | Branch_ID | Date_Joined  | Contact
-------------------------------------------------------------------------------
3            | Rajesh Kumar | Wing Commander | 3       | 2006-06-11   | 9876543230
2            | Suresh Mehra | Commander      | 2       | 2007-05-20   | 9876543220
1            | Amit Singh   | Colonel        | 1       | 2005-03-14   | 9876543210
```

9. **Aggregate Functions**
   Performs calculations on a set of values and returns a single value. Common functions include:
   - `COUNT`: Counts the number of rows.
   - `SUM`: Adds values.
   - `AVG`: Returns the average value.
   - `MIN`/`MAX`: Returns the minimum or maximum value.

Input:

```
SELECT AVG(Quantity) AS Average_Equipment
FROM Equipment;
```

Output:

```
Average_Equipment
-----------------
351.67
```

10. **Pattern Matching (LIKE Operator)**
    Searches for a specified pattern in a column using wildcard characters:

- `%`: Matches any number of characters.
- `_`: Matches a single character.

Input:

```
SELECT * FROM Defense_Personnel
WHERE Name LIKE 'A%';
```

Output:

```
Personnel_ID | Name         | Rank     | Branch_ID | Date_Joined   | Contact
-------------------------------------------------------------------------
1            | Amit Singh   | Colonel  | 1         | 2005-03-14    | 9876543210
4            | Aman Joshi   | Major    | 2         | 2012-08-20    | 9876543250
```

11. Nested Queries and Subqueries

A query inside another query. Subqueries can be used in `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statements. They are often used to break down complex problems or perform comparisons.

Input:

```
-- Find personnel who joined after the average joining date
SELECT * FROM Defense_Personnel
WHERE Date_Joined > (SELECT AVG(Date_Joined) FROM Defense_Personnel);
```

Output:

```
Personnel_ID | Name          | Rank            | Branch_ID | Date_Joined   | Contact
-----------------------------------------------------------------------------
2            | Suresh Mehra  | Commander       | 2         | 2007-05-20    | 9876543220
3            | Rajesh Kumar  | Wing Commander  | 3         | 2006-06-11    | 9876543230
```

12. Set Operations

Combines the results of two or more queries. Common operations include:

- **UNION**: Combines the result sets of two queries and removes duplicates.
- **INTERSECT**: Returns only rows that appear in both result sets.
- **EXCEPT**: Returns rows from the first query that are not in the second query.

Input:

```
-- Union of two select queries
SELECT Name FROM Defense_Personnel
WHERE Branch_ID = 1
UNION
SELECT Name FROM Defense_Personnel
WHERE Branch_ID = 2;
```

Output:

```
- Union
Name
--------------
Amit Singh
Suresh Mehra
```

Conclusion

This project on **Indian Defense Services** demonstrates the comprehensive application of database management concepts using SQL. Through the implementation of **DDL**, **DML**, **TCL**, and **VDL** commands, we created and manipulated tables representing different entities such as branches, personnel, and equipment

within the defense services. We also enforced data integrity using constraints like **Primary Keys**, **Foreign Keys**, and **Not Null** to maintain the consistency and accuracy of the data.

Various types of **Joins** were applied to showcase how different tables are connected and to retrieve meaningful insights from the data. Advanced querying techniques such as **Nested Queries**, **Subqueries**, and **Set Operations** allowed us to explore complex relationships between the entities. Additionally, by utilizing **Aggregate Functions**, **Group By**, and **Having Clauses**, we aggregated and filtered data for detailed analysis. The project also incorporated essential transaction control commands to manage database integrity during data manipulation.

Through this project, we've demonstrated a real-world implementation of SQL in managing structured data for a domain as complex and crucial as Indian defense services. It highlights the importance of robust database design and querying for efficient data management and retrieval in any domain.