Database Manager in Bash - Full Explanation

Introduction

This document explains in detail how the Bash-based Database Manager works. It includes an expla

Program Flow

1. The script starts by showing a main menu.
2. User chooses an option (Create, Insert, Update, Exit).
3. Based on the choice, the corresponding function runs.
4. Schema and data are stored in plain text files inside a directory for each table.
5. AWK and Bash arrays are used to parse and update data.

Schema Handling

Each table has a schema file in the format:

```
columns:id|name|salary
types:number|string|number
primary:id
unique:id
notnull:id|name
```

This schema is read into arrays:
- DB_COLUMNS = column names
- DB_TYPES = data types (number/string)
- DB_PRIMARY_KEY = primary key
- DB_UNIQUE_COLS = unique columns
- DB_NOTNULL_COLS = not null columns

How Data is Stored

Rows are stored in a CSV-like format inside a data file:

```
1,Hima,500
2,Ahmed,600
3,Mohamed,700
```

Each value corresponds to the schema columns in order. Data is appended line by line.

Insert Logic

1. Schema is loaded using read_schema().
2. For each column:
   - User is asked to enter a value.
   - If column is NOT NULL → must not be empty.
   - If type=number → must match regex for numeric values.
   - If UNIQUE → check existing rows with cut + grep.
3. Once validated, values are joined with commas and appended to the data file.

Search / WHERE Conditions

```
WHERE conditions are parsed using regex:
Example: id<=2
Regex extracts:
- Column = id
- Operator = <=
- Value = 2

AWK is used to apply the condition to each row. For example:

awk -F',' -v pos=1 -v val=2 '$pos <= val' datafile

This returns all rows where the first column is <= 2.
```

Update Logic

```
Update uses both WHERE condition and SET expression.

1. Ask for WHERE condition → parse column, operator, value.
2. Ask for SET column and new value.
3. AWK scans all rows:
   - If condition is true, update column value.
   - Otherwise keep row unchanged.
4. Supports:
   - Direct replacement (value=200)
   - Increment (+N)
   - Decrement (-N)

Example:
WHERE id>=3 SET salary=+100
→ increases salary by 100 for all rows with id >= 3.
```

Main Loop

```
The main loop keeps showing the menu until the user chooses Exit.
It calls the correct function based on choice using a case statement.
```

Bash Syntax Used

```
- Functions: function_name() { ... }
- Arrays: DB_COLUMNS=(id name)
- Array expansion: ${DB_COLUMNS[@]}
- Regex in Bash: [[ string =~ regex ]]
- Arithmetic: $((i+1))
- AWK: awk -F',' '{ ... }' file
- ANSI Colors: \033[31m (red), \033[32m (green)
- IFS: change input field separator for splitting strings
- Here-string: <<< feeds string into command
```

Summary

```
This Bash Database Manager mimics simple SQL features:
- CREATE TABLE with schema
- INSERT with constraints
- UPDATE with conditions and arithmetic
- Data stored as text files
- Search performed with AWK
```

It demonstrates how Bash can implement database-like logic using text files and Unix tools.