Aim: Implementation of Dimension and Fact tables and perform OLAP operations.

Objective: OLAP stands for Online Analytical Processing. The objective of OLAP is to analyze information from multiple database systems at the same time. It is based on multidimensional data model and allows the user to query on multi-dimensional data.

Theory:

- Online Analytical Processing Server (OLAP) is based on the multidimensional data model.
- The main aim of OLAP is to provide multidimensional analysis to the underlying data.

  Following is the list of OLAP operations:

  1. Roll-up
  2. Drill-down
  3. Slice
  4. Dice
  5. Pivot (rotate)

Roll-up:

- The roll-up operation (also called the drill-up operation) performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction.
- Figure 2.1 shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for location.
- This hierarchy was defined as the total order "street < city < province or state < country."
- The roll-up operation aggregates the data by ascending the location hierarchy from the level of city to the level of country.
- In other words, rather than grouping the data by city, the resulting cube groups the data by country.

Drill-down:

- Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data.
- Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.
- Figure 2.1 shows the result of a drill-down operation performed on the central cube by stepping down a concept hierarchy for time defined as "day < month < quarter < year."
- Drill-down occurs by descending the time hierarchy from the level of quarter to the more detailed level of month.
- The resulting data cube details the total sales per month rather than summarizing them by quarter.

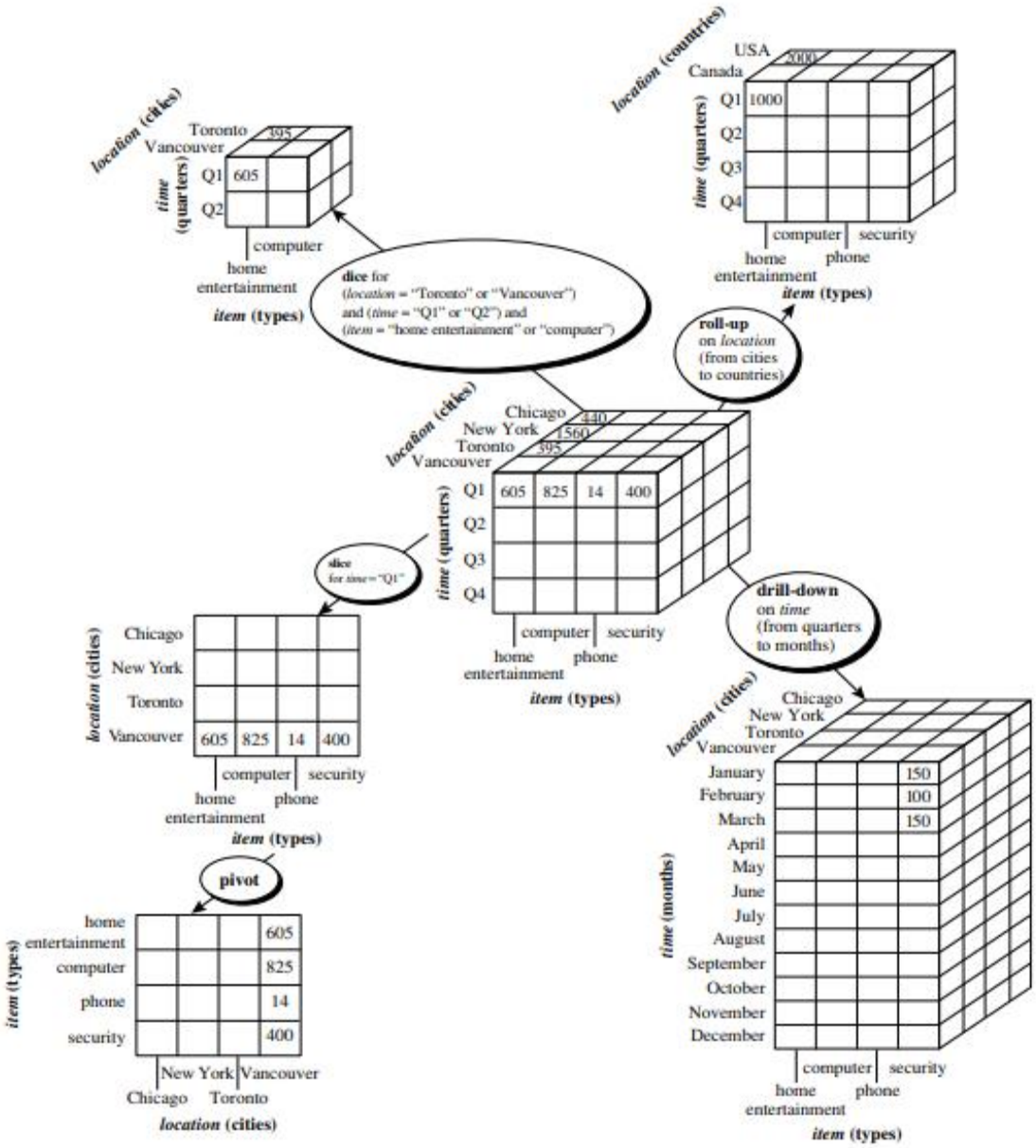


Figure 2.1: Examples of typical OLAP operations on multidimensional data.

Slice:

- The slice operation performs a selection on one dimension of the given cube, resulting in a subcube.
- Figure 2.1 below shows a slice operation where the sales data are selected from the central cube for the dimension time using the criterion time = "Q1."

Dice:

- The dice operation defines a subcube by performing a selection on two or more dimensions.
- Figure 2.1 shows a dice operation on the central cube based on the following selection criteria that involve three dimensions: (location = "Toronto" or "Vancouver") and (time = "Q1" or "Q2") and (item = "home entertainment" or "computer").

Pivot:

- Pivot (also called rotate) is a visualization operation that rotates the data axes in view to provide an alternative data presentation.
- Figure 2.1 shows a pivot operation where the item and location axes in a 2-D slice are rotated.

Problem Statement:

The problem is to design and implement a data warehousing solution for a bookstore that optimally organizes and manages its vast data, including sales, inventory, customer information, and more, to facilitate efficient reporting and analytics. This involves creating both a star schema and a snowflake schema to support various business intelligence and decision-making processes, while ensuring data accuracy, integrity, and performance

Code:

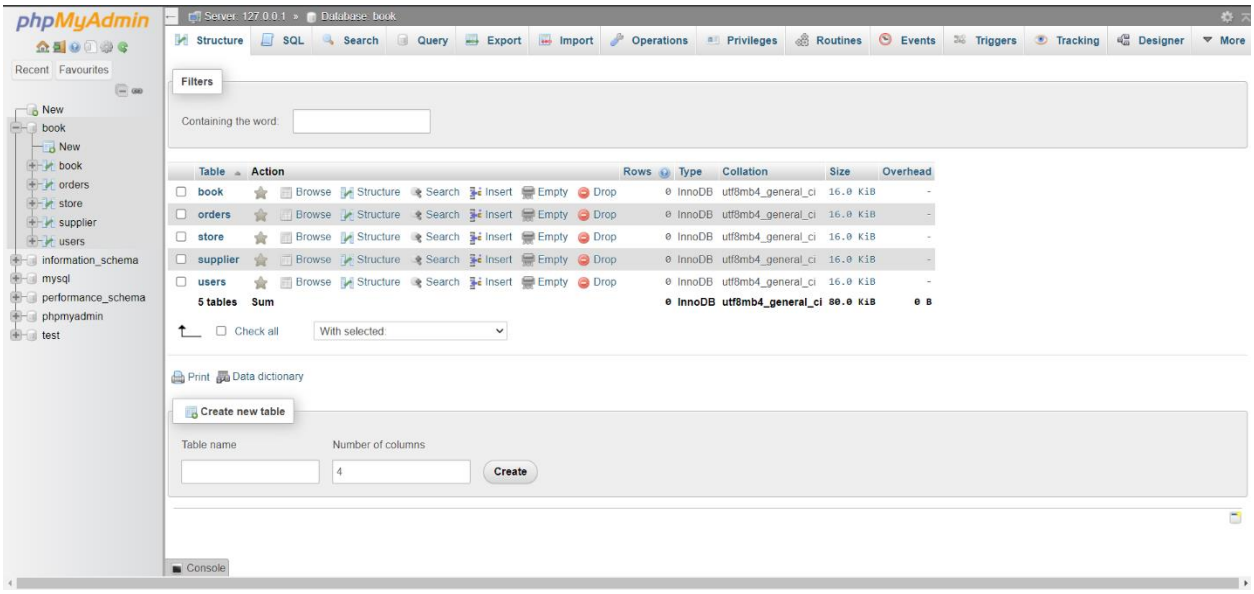
Create Database "book":

```
Create database book;
```

Create Dimension Tables:

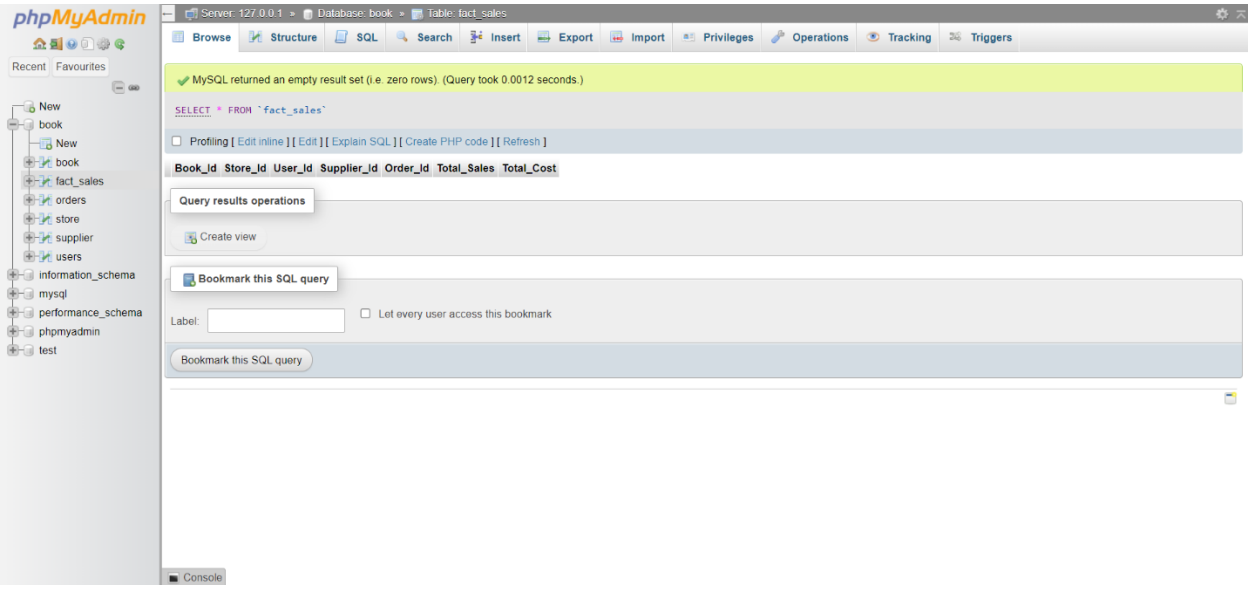
```
CREATE TABLE Book (
  Book_Id INT PRIMARY KEY,
  Book_Name VARCHAR(100),
  Book_Genre VARCHAR(50),
  Book_Author VARCHAR(100),
  Book_Cost INT
);

CREATE TABLE Store (
   Store_Id INT PRIMARY KEY,
   Store_Name VARCHAR(100),
   Store_Address VARCHAR(200),
   Store_Pincode INT,
   Store_City VARCHAR(100)
);
```

```
CREATE TABLE Supplier (
  Supplier_Id INT PRIMARY KEY,
  Supplier_Name VARCHAR(100),
  Supplier_Address VARCHAR(200)
);

CREATE TABLE Users (
  User_Id INT PRIMARY KEY,
  User_Name VARCHAR(100),
  User_Address VARCHAR(200),
  User_Pincode INT,
  User_City VARCHAR(100)
);

CREATE TABLE Orders (
  Order_Id INT PRIMARY KEY,
  Order_Cost INT,
  Order_Quantity INT
);
```



Create Fact Table:

```
CREATE TABLE Fact_Sales (
  Book_Id INT,
  Store_Id INT,
  User_Id INT,
  Supplier_Id INT,
  Order_Id INT,
  Total_Sales INT,
  Total_Cost INT,
  PRIMARY KEY (Book_Id, Store_Id, Order_Id),
  FOREIGN KEY (Book_Id) REFERENCES Book(Book_Id),
  FOREIGN KEY (Store_Id) REFERENCES Store(Store_Id),
  FOREIGN KEY (User_Id) REFERENCES Users(User_Id),
  FOREIGN KEY (Supplier_Id) REFERENCES Supplier(Supplier_Id),
  FOREIGN KEY (Order_Id) REFERENCES Orders(Order_Id)
);
```



Inserting values in both dimension and fact tables:

```
INSERT INTO Book VALUES
  (1, 'To Kill a Mockingbird', 'Fiction', 'Harper Lee', 250),
  (2, '1984', 'Fiction', 'George Orwell', 200),
  (3, 'The Great Gatsby', 'Fiction', 'F. Scott Fitzgerald', 180);

INSERT INTO Store VALUES
  (1, 'A Store', '123 Main St', 12345, 'New York'),
  (2, 'B Store', '456 Oak Rd', 67890, 'Chicago'),
  (3, 'C Store', '789 Elm St', 23456, 'Los Angeles');

INSERT INTO Supplier VALUES
  (1, 'ABC Supplier', '111 Industry Ave'),
  (2, 'XYZ Supplier', '222 Commerce St'),
  (3, '123 Supplier', '333 Trade Blvd');

INSERT INTO Users VALUES
  (1, 'John Doe', '100 1st St', 12345, 'New York'),
  (2, 'Jane Smith', '200 2nd St', 23456, 'Los Angeles'),
  (3, 'Bob Johnson', '300 3rd St', 34567, 'Chicago');

INSERT INTO Orders VALUES
  (1, 500, 10),
  (2, 400, 8),
  (3, 600, 12);

INSERT INTO Fact_Sales VALUES
  (1, 1, 1, 1, 1, 5000, 2500),
  (2, 2, 2, 2, 2, 4000, 1600),
  (3, 3, 3, 3, 3, 7200, 2160);
```

Displaying the tables:

SELECT * FROM book;

```
SELECT * FROM book;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	Book_Id	Book_Name	Book_Genre	Book_Author	Book_Cost
Edit Copy Delete	1	To Kill a Mockingbird	Fiction	Harper Lee	250
Edit Copy Delete	2	1984	Fiction	George Orwell	200
Edit Copy Delete	3	The Great Gatsby	Fiction	F. Scott Fitzgerald	180

SELECT * FROM fact_sales;

```
SELECT * FROM fact_sales;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	Book_Id	Store_Id	User_Id	Supplier_Id	Order_Id	Total_Sales	Total_Cost
Edit Copy Delete	1	1	1	1	1	5000	2500
Edit Copy Delete	2	2	2	2	2	4000	1600
Edit Copy Delete	3	3	3	3	3	7200	2160

SELECT * FROM orders;

```
SELECT * FROM orders;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	Order_Id	Order_Cost	Order_Quantity
Edit Copy Delete	1	500	10
Edit Copy Delete	2	400	8
Edit Copy Delete	3	600	12

SELECT * FROM store;

```
SELECT * FROM store;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	Store_Id	Store_Name	Store_Address	Store_Pincode	Store_City
Edit Copy Delete	1	A Store	123 Main St	12345	New York
Edit Copy Delete	2	B Store	456 Oak Rd	67890	Chicago
Edit Copy Delete	3	C Store	789 Elm St	23456	Los Angeles

SELECT * FROM supplier;

```
SELECT * FROM supplier;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	Supplier_Id	Supplier_Name	Supplier_Address
Edit Copy Delete	1	ABC Supplier	111 Industry Ave
Edit Copy Delete	2	XYZ Supplier	222 Commerce St
Edit Copy Delete	3	123 Supplier	333 Trade Blvd

SELECT * FROM users;

```
SELECT * FROM users;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	User_Id	User_Name	User_Address	User_Pincode	User_City
Edit Copy Delete	1	John Doe	100 1st St	12345	New York
Edit Copy Delete	2	Jane Smith	200 2nd St	23456	Los Angeles
Edit Copy Delete	3	Bob Johnson	300 3rd St	34567	Chicago

Write SQL Queries for all the above OLAP operations:

Rollup:

SELECT Book_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost
FROM Fact_Sales
GROUP BY Book_Id;

```
SELECT Book_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost FROM Fact_Sales GROUP BY Book_Id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	Book_Id	Total_Sales	Total_Cost
Edit Copy Delete	1	5000	2500
Edit Copy Delete	2	4000	1600
Edit Copy Delete	3	7200	2160

Drill Down:

SELECT Book_Id, Store_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost
FROM Fact_Sales
GROUP BY Book_Id, Store_Id;

```
SELECT Book_Id, Store_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost FROM Fact_Sales GROUP BY Book_Id, Store_Id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	Book_Id	Store_Id	Total_Sales	Total_Cost
Edit Copy Delete	1	1	5000	2500
Edit Copy Delete	2	2	4000	1600
Edit Copy Delete	3	3	7200	2160

Slice:

SELECT Book_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost
FROM Fact_Sales f
JOIN User u ON f.User_Id = u.User_Id
WHERE u.User_City IN ('New York', 'Los Angeles')
GROUP BY f.Book_Id;

```
SELECT Book_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost FROM Fact_Sales f JOIN Users u ON f.User_Id = u.User_Id WHERE u.User_City IN ('New York', 'Los Angeles') GROUP BY f.Book_Id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Book_Id	Total_Sales	Total_Cost
1	5000	2500
2	4000	1600

Dice:

SELECT Store_Id, Supplier_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost
FROM Fact_Sales
GROUP BY Store_Id, Supplier_Id;

```
SELECT Store_Id, Supplier_Id, SUM(Total_Sales) AS Total_Sales, SUM(Total_Cost) AS Total_Cost FROM Fact_Sales GROUP BY Store_Id, Supplier_Id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	Store_Id	Supplier_Id	Total_Sales	Total_Cost
Edit Copy Delete	1	1	5000	2500
Edit Copy Delete	2	2	4000	1600
Edit Copy Delete	3	3	7200	2160

Pivot:

```
SELECT
  User_City,
  SUM(CASE WHEN Book_Id = 1 THEN Total_Sales END) AS Book_1_Sales,
  SUM(CASE WHEN Book_Id = 1 THEN Total_Cost END) AS Book_1_Cost,
  SUM(CASE WHEN Book_Id = 2 THEN Total_Sales END) AS Book_2_Sales,
  SUM(CASE WHEN Book_Id = 2 THEN Total_Cost END) AS Book_2_Cost,
  SUM(CASE WHEN Book_Id = 3 THEN Total_Sales END) AS Book_3_Sales,
  SUM(CASE WHEN Book_Id = 3 THEN Total_Cost END) AS Book_3_Cost
FROM Fact_Sales f
JOIN User u ON f.User_Id = u.User_Id
GROUP BY User_City;
```

```
SELECT User_City, SUM(CASE WHEN Book_Id = 1 THEN Total_Sales END) AS Book_1_Sales, SUM(CASE WHEN Book_Id = 1 THEN Total_Cost END) AS Book_1_Cost, SUM(CASE WHEN Book_Id = 2 THEN Total_Sales END) AS Book_2_Sales, SUM(CASE WHEN Book_Id = 2 THEN Total_Cost END) AS Book_2_Cost, SUM(CASE WHEN Book_Id = 3 THEN Total_Sales END) AS Book_3_Sales, SUM(CASE WHEN Book_Id = 3 THEN Total_Cost END) AS Book_3_Cost FROM Fact_Sales f JOIN Users u ON f.User_Id = u.User_Id GROUP BY User_City;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

User_City	Book_1_Sales	Book_1_Cost	Book_2_Sales	Book_2_Cost	Book_3_Sales	Book_3_Cost
Chicago	NULL	NULL	NULL	NULL	7200	2160
Los Angeles	NULL	NULL	4000	1600	NULL	NULL
New York	5000	2500	NULL	NULL	NULL	NULL

Conclusion:

Thus, we have learned implementation of Dimension and Fact tables and perform OLAP operations on database. OLAP stands for Online Analytical Processing. The objective of OLAP is to analyze information from multiple database systems at the same time.