

LAB Manual  
PART A  
(PART A : TO BE REFERRED BY STUDENTS)

## Experiment No.03

### A.1 Aim:

Implementation of OLAP operation in SQL Environment.

### A.2 Prerequisite:

Refer the DBMS manual for SQL Commands and ER diagram.

### A.3 Outcome:

After successful completion of this experiment students will be able to

- ☐ Apply various OLAP operations and also able to describe the data mining concept and techniques.

### A.4 Theory:

On-Line Analytical Processing (OLAP) is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access in a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user.

**OLAP Models:-**(Detailed Description is required for following models)

- MOLAP
- ROLAP

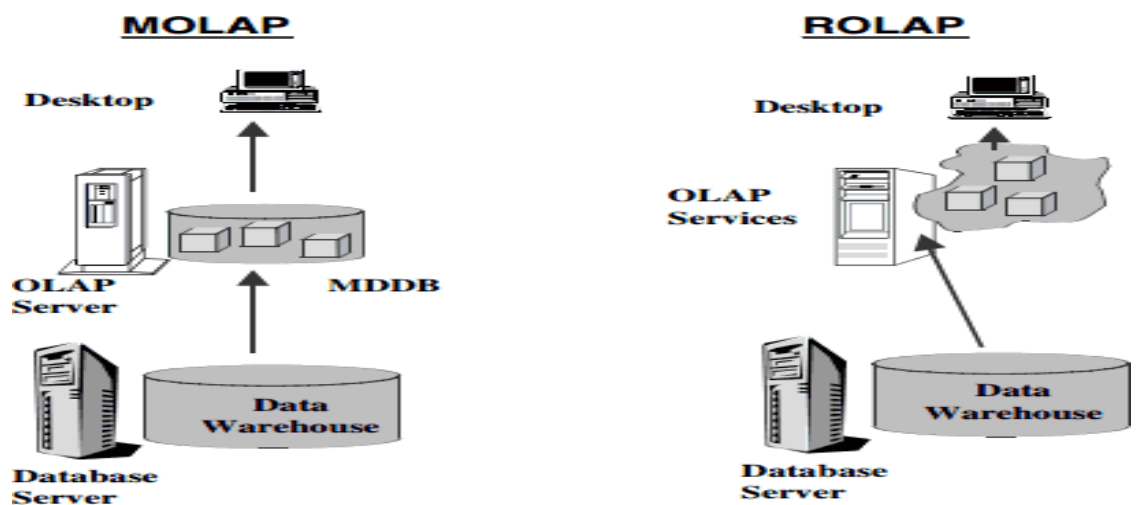


Figure 15-15 OLAP models.

## OLAP Operations:

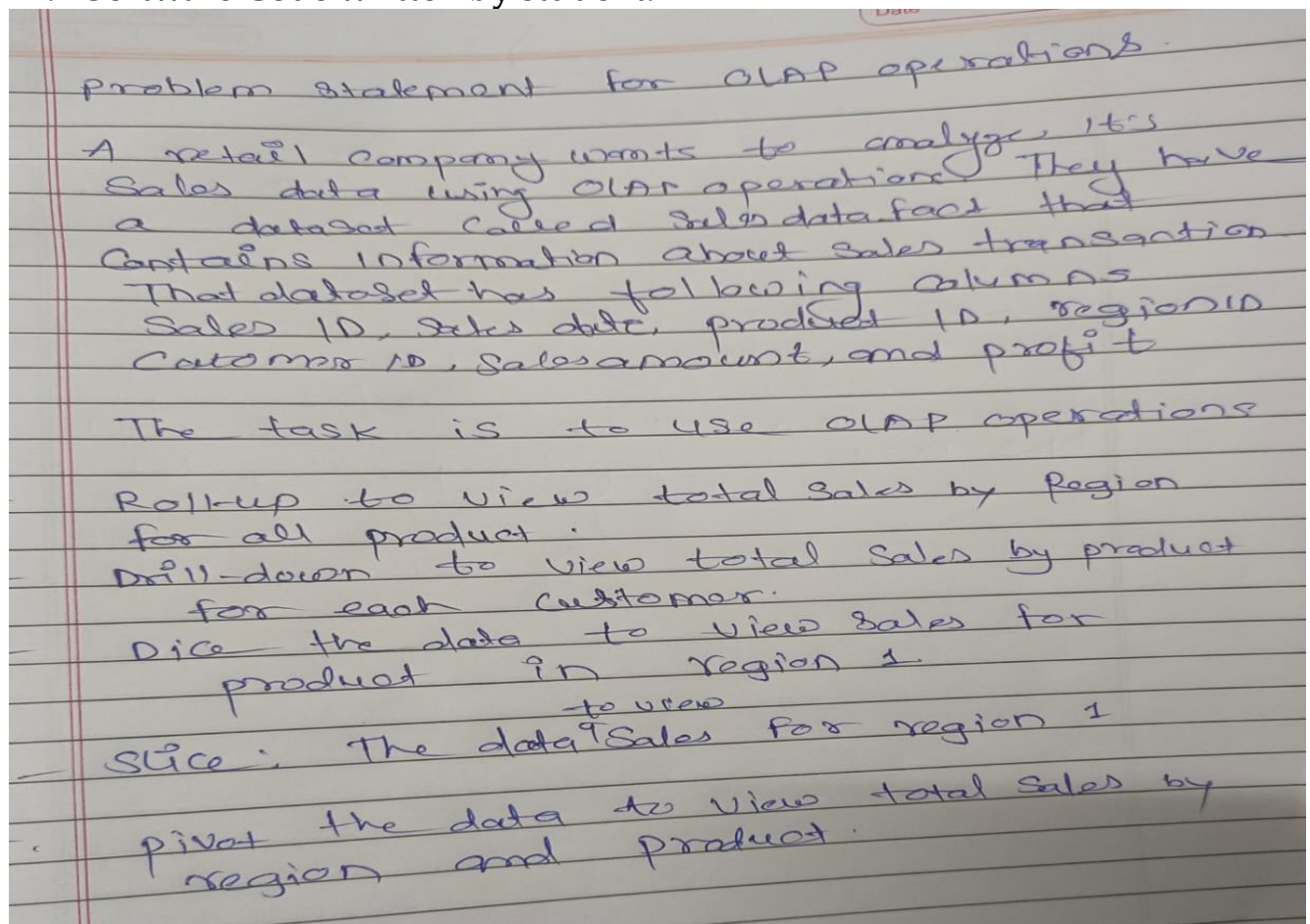
- Roll up (drill-up): summarize data
  - by climbing up hierarchy or by dimension reduction
- Drill down (roll down): reverse of roll-up
  - from higher level summary to lower level summary or detailed data, or introducing new dimensions
- Slice and dice: project and select
- Pivot (rotate):
  - reorient the cube, visualization, 3D to series of 2D planes
- Other operations
  - drill across: involving (across) more than one fact table
  - drill through: through the bottom level of the cube to its back-end relational tables (using SQL)

## PART B

(PART B : TO BE COMPLETED BY STUDENTS)

Roll. No. A12	Name:SUFYAN KHAN
Class: T.E.	Batch:A1
Experiment No:3	Date of Experiment:31/07/2023
Date of Submission :07/08/2023	Grade:

### B.1 Software Code written by student:



*Creation of Table:*

```
CREATE TABLE Products (  
    ProductID INT PRIMARY KEY,  
    ProductName VARCHAR(50)  
);
```

```
CREATE TABLE Regions (  
    RegionID INT PRIMARY KEY,  
    RegionName VARCHAR(50)  
);
```

```
CREATE TABLE Customers (  
    CustomerID INT PRIMARY KEY,  
    CustomerName VARCHAR(50)  
);
```

```
CREATE TABLE SalesData_Fact (  
    SaleID INT PRIMARY KEY,  
    SalesDate DATE,  
    ProductID INT,  
    RegionID INT,  
    CustomerID INT,  
    SalesAmount DECIMAL(10, 2),  
    Profit DECIMAL(10, 2),  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),  
    FOREIGN KEY (RegionID) REFERENCES Regions(RegionID),  
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);
```

*Insertion of Value in tables:*

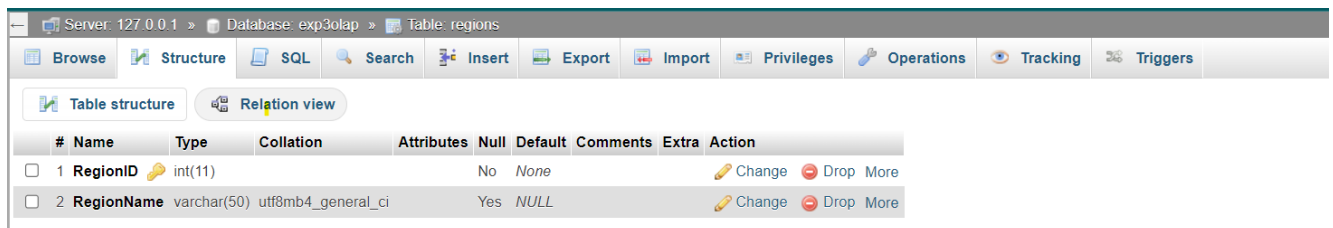
```
INSERT INTO Products (ProductID, ProductName)  
VALUES  
  (1, 'ProductA'),  
  (2, 'ProductB'),  
  (3, 'ProductC');
```



The screenshot shows the phpMyAdmin interface for the 'products' table. The 'Table structure' tab is active, displaying the following table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ProductID	int(11)			No	None			Change Drop More
2	ProductName	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

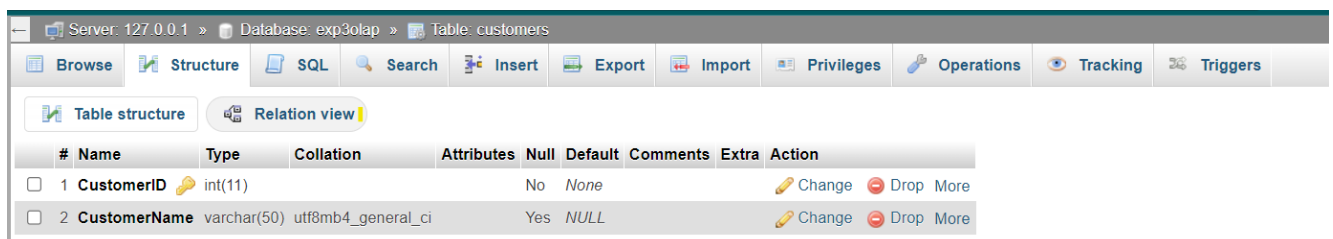
```
INSERT INTO Regions (RegionID, RegionName)  
VALUES  
  (1, 'Region1'),  
  (2, 'Region2');
```



The screenshot shows the phpMyAdmin interface for the 'regions' table. The 'Table structure' tab is active, displaying the following table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	RegionID	int(11)			No	None			Change Drop More
2	RegionName	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

```
INSERT INTO Customers (CustomerID, CustomerName)  
VALUES  
  (1, 'Customer1'),  
  (2, 'Customer2'),  
  (3, 'Customer3');
```



The screenshot shows the phpMyAdmin interface for the 'customers' table. The 'Table structure' tab is active, displaying the following table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	CustomerID	int(11)			No	None			Change Drop More
2	CustomerName	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

```

INSERT INTO SalesData_Fact (SaleID, SalesDate, ProductID, RegionID, CustomerID,
SalesAmount, Profit)
VALUES
(1, '2023-07-01', 1, 1, 1, 1000.00, 200.00),
(2, '2023-07-02', 2, 2, 2, 500.00, 100.00),
(3, '2023-07-03', 1, 1, 3, 800.00, 150.00),
(4, '2023-07-04', 3, 2, 1, 1200.00, 300.00),
(5, '2023-07-05', 1, 1, 2, 600.00, 120.00),
(6, '2023-07-06', 2, 2, 3, 700.00, 140.00);

```

The screenshot shows a database management interface with the following table structure for 'salesdata\_fact':

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	SaleID	int(11)			No	None			Change Drop More
2	SalesDate	date			Yes	NULL			Change Drop More
3	ProductID	int(11)			Yes	NULL			Change Drop More
4	RegionID	int(11)			Yes	NULL			Change Drop More
5	CustomerID	int(11)			Yes	NULL			Change Drop More
6	SalesAmount	decimal(10,2)			Yes	NULL			Change Drop More
7	Profit	decimal(10,2)			Yes	NULL			Change Drop More

## B.1 Input and Output:

**Input :**

SQL commands/script

**RollUp:**

```

SELECT ProductID, SUM(SalesAmount) AS TotalSales, SUM(Profit) AS TotalProfit,
       (SUM(Profit) / SUM(SalesAmount)) * 100 AS ProfitMargin
FROM SalesData_Fact
GROUP BY ProductID;

```

**Drill-Down:**

```

SELECT CustomerID, ProductID, SUM(SalesAmount) AS TotalSales
FROM SalesData_Fact
GROUP BY CustomerID, ProductID;

```

**Dice:**

```

SELECT SalesDate, SalesAmount
FROM SalesData_Fact
WHERE ProductID = 1 AND RegionID = 1;

```

**Slice:**

```

SELECT SalesDate, ProductID, SalesAmount

```

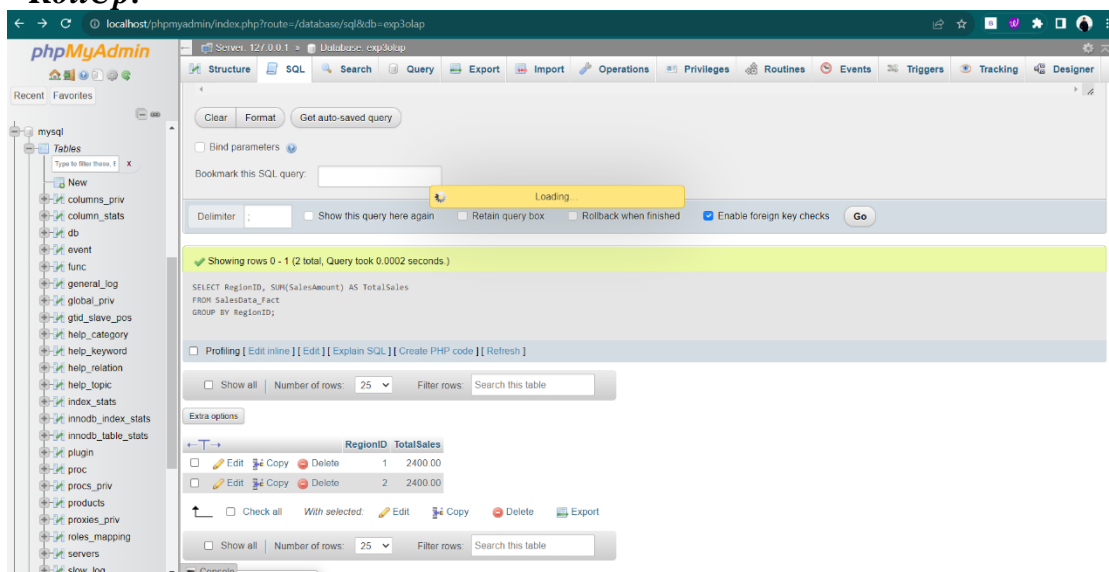
***FROM SalesData\_Fact  
WHERE RegionID = 1;***

***Pivot:***

***SELECT RegionID,  
MAX(CASE WHEN ProductID = 1 THEN SalesAmount ELSE NULL END) AS  
ProductA\_Sales,  
MAX(CASE WHEN ProductID = 2 THEN SalesAmount ELSE NULL END) AS  
ProductB\_Sales,  
MAX(CASE WHEN ProductID = 3 THEN SalesAmount ELSE NULL END) AS  
ProductC\_Sales  
FROM SalesData\_Fact  
GROUP BY RegionID;***

**Output:**

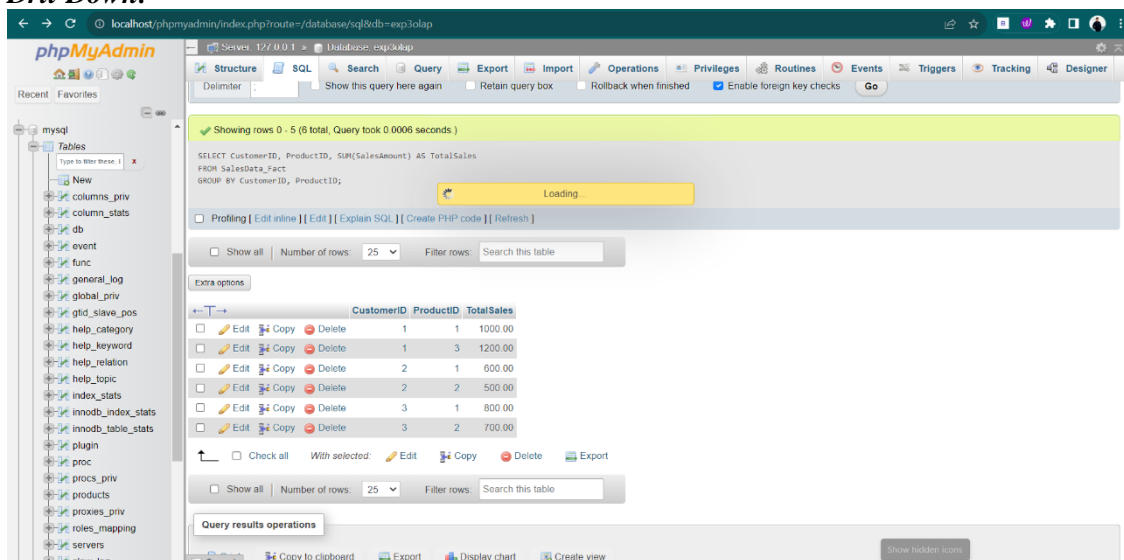
***RollUp:***



The screenshot shows the phpMyAdmin interface with a SQL query executed. The query is: `SELECT RegionID, SUM(SalesAmount) AS TotalSales FROM SalesData_Fact GROUP BY RegionID;` The result set shows two rows for RegionID 1 and 2, both with a TotalSales of 2400.00.

RegionID	TotalSales
1	2400.00
2	2400.00

***Drill-Down:***



The screenshot shows the phpMyAdmin interface with a SQL query executed. The query is: `SELECT CustomerID, ProductID, SUM(SalesAmount) AS TotalSales FROM SalesData_Fact GROUP BY CustomerID, ProductID;` The result set shows six rows for CustomerID 1, 2, and 3, grouped by ProductID. The TotalSales values are 1000.00, 1200.00, 600.00, 500.00, 800.00, and 700.00 respectively.

CustomerID	ProductID	TotalSales
1	1	1000.00
1	3	1200.00
2	1	600.00
2	2	500.00
3	1	800.00
3	2	700.00

## Dice:

The screenshot shows the phpMyAdmin interface with the 'exp3olap' database selected. The 'SQL' tab is active, displaying a query that filters for ProductID = 1 and RegionID = 1. The result shows 2 rows of sales data.

```
SELECT SalesDate, SalesAmount
FROM SalesData_Fact
WHERE ProductID = 1 AND RegionID = 1;
```

SalesDate	SalesAmount
2023-07-01	1000.00
2023-07-03	800.00

## Slice:

The screenshot shows the phpMyAdmin interface with the 'exp3olap' database selected. The 'SQL' tab is active, displaying a query that filters for RegionID = 1. The result shows 3 rows of sales data for different products.

```
SELECT SalesDate, ProductID, SalesAmount
FROM SalesData_Fact
WHERE RegionID = 1;
```

SalesDate	ProductID	SalesAmount
2023-07-01	1	1000.00
2023-07-03	1	800.00
2023-07-05	1	600.00

## Pivot:

The screenshot shows the phpMyAdmin interface with the 'mysql' database selected. The 'SQL' tab is active, displaying a complex pivot query. The result shows 2 rows of aggregated sales data by RegionID.

```
SELECT RegionID,
MAX(CASE WHEN ProductID = 1 THEN SalesAmount ELSE NULL END) AS ProductA_Sales,
MAX(CASE WHEN ProductID = 2 THEN SalesAmount ELSE NULL END) AS ProductB_Sales,
MAX(CASE WHEN ProductID = 3 THEN SalesAmount ELSE NULL END) AS ProductC_Sales
FROM SalesData_Fact
GROUP BY RegionID;
```

RegionID	ProductA_Sales	ProductB_Sales	ProductC_Sales
1	1000.00	NULL	NULL
2	NULL	700.00	1200.00



## **B.2 Observations and learning:**

From this practical, we learned about OLAP (Online Analytical Processing) operations and how they enable us to analyze data from different dimensions and perspectives. The creation of the SalesData\_Fact table with foreign key references to dimension tables (Products, Regions, and Customers) allowed us to perform OLAP operations effectively. The use of SQL queries for slicing, dicing, pivoting, drill-down, and roll-up provided valuable insights into the sales data, enabling us to analyze it at various levels of granularity. Additionally, we learned how to calculate the profit margin for each product, which is a crucial metric for business analysis. Overall, this practical demonstrated the power of OLAP in data analysis and its importance in making informed business decisions.

## **B.3 Conclusion:**

*In conclusion, the experiment focused on understanding OLAP operations and their practical application in data analysis. Through the creation of the SalesData\_Fact table and employing SQL queries for various OLAP operations, we gained valuable insights into the sales data from multiple dimensions. These operations allowed us to view, analyze, and drill down into data efficiently, providing a comprehensive understanding of the dataset. Overall, the experiment highlighted the significance of OLAP in enabling businesses to make data-driven decisions and uncover valuable patterns and trends.*



## B.4 Question of Curiosity

Q1: Explain Need for Online Analytical Processing.

OLAP is essential for multidimensional analysis, providing faster query responses and enabling dynamic data exploration. It empowers decision-makers with real-time insights, supporting better-informed business decisions and fostering a data-driven culture within organizations. OLAP's pre-aggregated data storage and scalability make it ideal for handling large datasets and complex analytical queries efficiently. Furthermore, OLAP plays a crucial role in business intelligence and decision support systems, facilitating ad-hoc analysis and report generation without relying on IT teams. Its flexibility and interactivity allow users to drill down into data, uncovering trends and patterns, while its efficient data aggregation and processing capabilities help manage large volumes of data effectively. Overall, OLAP is vital for transforming raw data into actionable insights, driving strategic planning, and improving organizational performance.

Q2: What are the differences between MOLAP and ROLAP?

**MOLAP:**

1. Pre-Aggregation: MOLAP systems store pre-aggregated data in multidimensional cubes, allowing for fast query response times. Aggregations are computed during cube creation, reducing the need for complex calculations during query execution.
2. Multidimensional Data Model: MOLAP leverages a multidimensional data model, where data is organized along various dimensions and hierarchies. This enables users to analyze data from different perspectives, providing deeper insights into business performance.

**ROLAP:**

1. On-the-Fly Aggregation: ROLAP systems perform aggregations on-the-fly using SQL queries when data is requested. Aggregated values are not pre-computed and rely on the underlying relational database for calculation.
2. Relational Data Model: ROLAP stores data in normalized relational databases, optimizing storage efficiency by reducing data redundancies. It offers greater flexibility in handling changes in data models and schema modifications.