**Practical No. 7**

# Title: OLAP WITH ORACLE

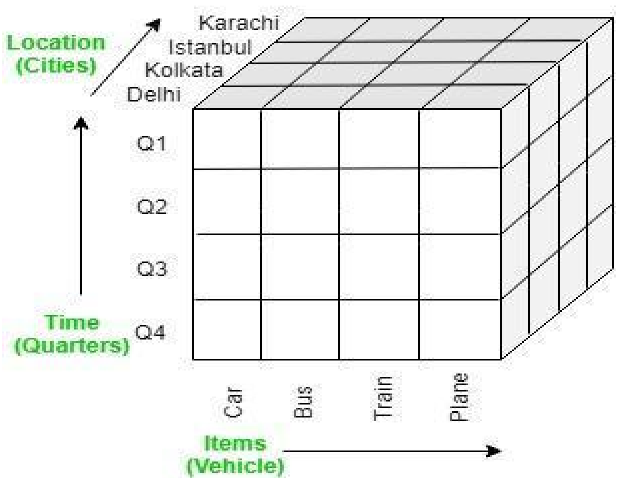
**Aim: : Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Rank AND Dense_rank.**

**Software required: Microsoft SQL Server OLAP Services version 7.0**

**Theory :-**

Oracle OLAP delivers advanced multidimensional analytic capabilities within Oracle Database .It is designed to provide excellent query performance, fast incremental updates of data sets, efficient management of summary data and rich analytic content. Oracle OLAP makes it easy to produce analytic measures, including time-series calculations, financial models, forecasts, allocations, regressions, and more. Hundreds of analytic functions can be easily combined in custom functions to solve nearly any analytic calculation requirement. Oracle OLAP cubes are represented using a star schema design: dimension views form a constellation around the cube (or fact) view. This standard representation of OLAP data makes it easy for any SQL-based tool or application to leverage the power of Oracle OLAP.

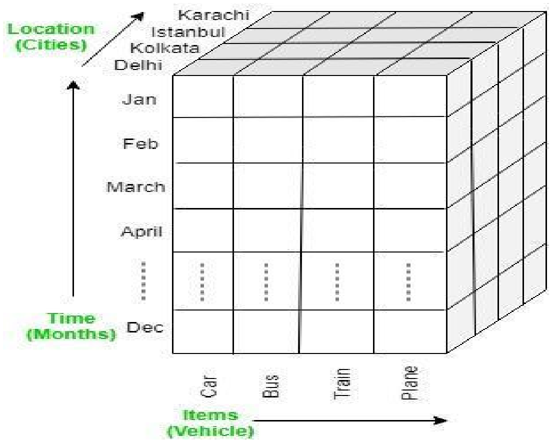
**OLAP** stands for **Online Analytical Processing** Server. It is a software technology that allows users to analyze information from multiple database systems at the same time. It is based on multidimensional data model and allows the user to query on multi-dimensional data (eg. Delhi -> 2018 -> Sales data). OLAP databases are divided into one or more cubes and these cubes are known as Hyper-cubes.



**OLAP operations:**

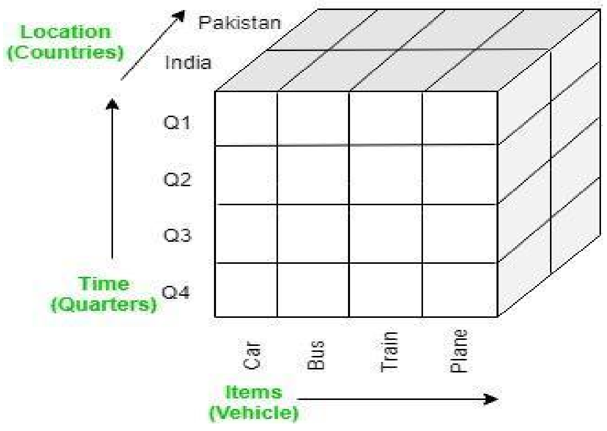
There are five basic analytical operations that can be performed on an OLAP cube:

1. **Drill down**: In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:
    - Moving down in the concept hierarchy
    - Adding a new dimension



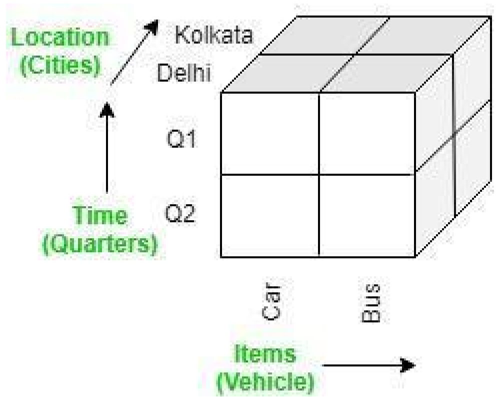
**2.Roll up:** It is just opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:

- Climbing up in the concept hierarchy
- Reducing the dimensions

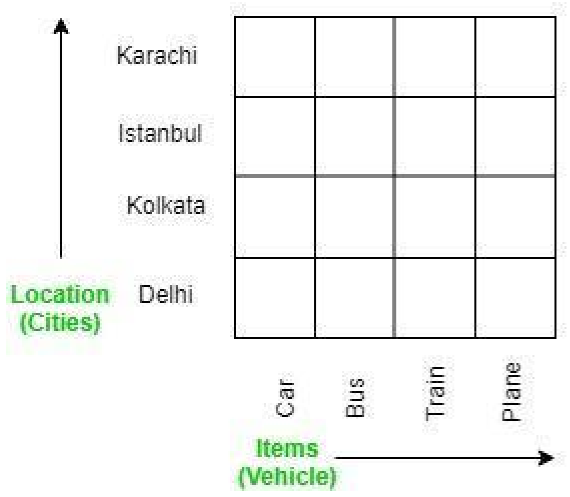


**3.Dice:** It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting following dimensions with criteria:

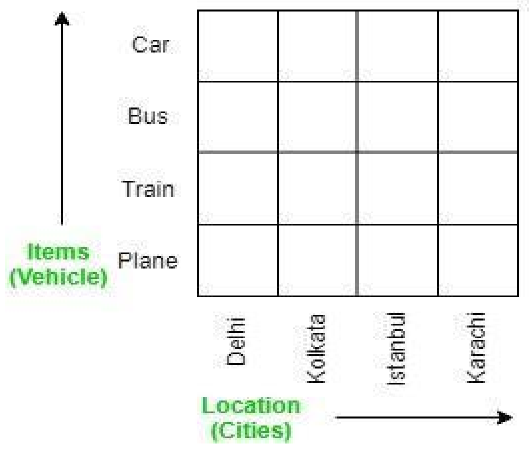
- Location = “Delhi” or “Kolkata”
- Time = “Q1” or “Q2”
- Item = “Car” or “Bus”



**4.Slice:** It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = “Q1”.



**5.Pivot:** It is also known as *rotation* operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing pivot operation gives a new view of it.



**Source Code :-**

let's assume we have a table named sales with the following structure:

**SQL :-**

```
  CREATE TABLE sales (

    product_id INT,

    category_id INT,

    sale_date DATE,

    amount DECIMAL(10, 2)

);


INSERT INTO sales VALUES

(1, 1, '2022-01-01', 100.00),

(2, 1, '2022-01-01', 150.00),
(1, 2, '2022-01-01', 200.00),

(2, 2, '2022-01-01', 120.00),

(1, 1, '2022-01-02', 80.00),

(2, 1, '2022-01-02', 180.00),

(1, 2, '2022-01-02', 220.00),

(2, 2, '2022-01-02', 100.00);
```

**ROLLUP Example:**

The ROLLUP operation is used to generate subtotals and grand totals. Here's an example:

```
SELECT category_id, sale_date, SUM(amount) AS total_sales

FROM sales

GROUP BY ROLLUP (category_id, sale_date);
```

**CUBE Example:**

The CUBE operation is used to generate subtotals and grand totals for all possible combinations of columns.

```
SELECT category_id, sale_date, SUM(amount) AS total_sales

FROM sales

GROUP BY CUBE (category_id, sale_date);
```

**FIRST and LAST Example:**

The FIRST and LAST functions can be used to get the first and last values in an ordered set.

```
-- First sale date for each category

SELECT category_id, MIN(sale_date) AS first_sale_date

FROM sales
```

```
GROUP BY category_id;

-- Last sale date for each category

SELECT category_id, MAX(sale_date) AS last_sale_date

FROM sales

GROUP BY category_id;
```

**RANK and DENSE_RANK Example:**

The RANK and DENSE_RANK functions are used to assign a rank to each row based on the specified order.

```
-- Rank products by total sales in descending order

SELECT product_id, SUM(amount) AS total_sales, RANK() OVER (ORDER BY
SUM(amount) DESC) AS sales_rank

FROM sales

GROUP BY product_id;
```

```
-- Dense rank products by total sales in descending order

SELECT product_id, SUM(amount) AS total_sales, DENSE_RANK() OVER (ORDER BY
SUM(amount) DESC) AS sales_dense_rank

FROM sales

GROUP BY product_id;
```

**Conclusion** :-

Implemented Analytical queries.

**FAQ -**

1) Explain the functionality of OLAP
2) List the applications of OLAP
3) What is a fact and Dimension table?
4) How do you query on OLAP Cube ?
5) What is the difference between slice and dice in OLAP?