Business Intelligence and Data Warehousing (ANL408)

- By Sabarish Nair

Recap from last week....

- OLAP
- OLAP Features
- OLAP Architecture
- MOLAP
- ROLAP
- HOLAP
- DOLAP
- Practical: Update Query
- Practical: Creating a temporary table
- Practical: Cleansing Text-based values
- Practical: Cleansing Numeric Values
- Practical: Creating a star-schema

Using a DWH – Common Use Cases

- Strategic Decision Making
- BI and Reporting
- Enable end users to analyze data
- Predictive Analytics
- Data-Driven Insights

Optimizing a Data Warehouse

What are indexes?

Types of Indices

Practical Guidelines

What are Indexes?

- Data structure used to improve the performance of the queries by enabling faster data retrieval.
- Created on columns that are more frequently used for:
  - Filtering
  - Joining
  - Sorting
- Full table scan is not required.
- Slower data writes and additional data storage

Types of Indexes

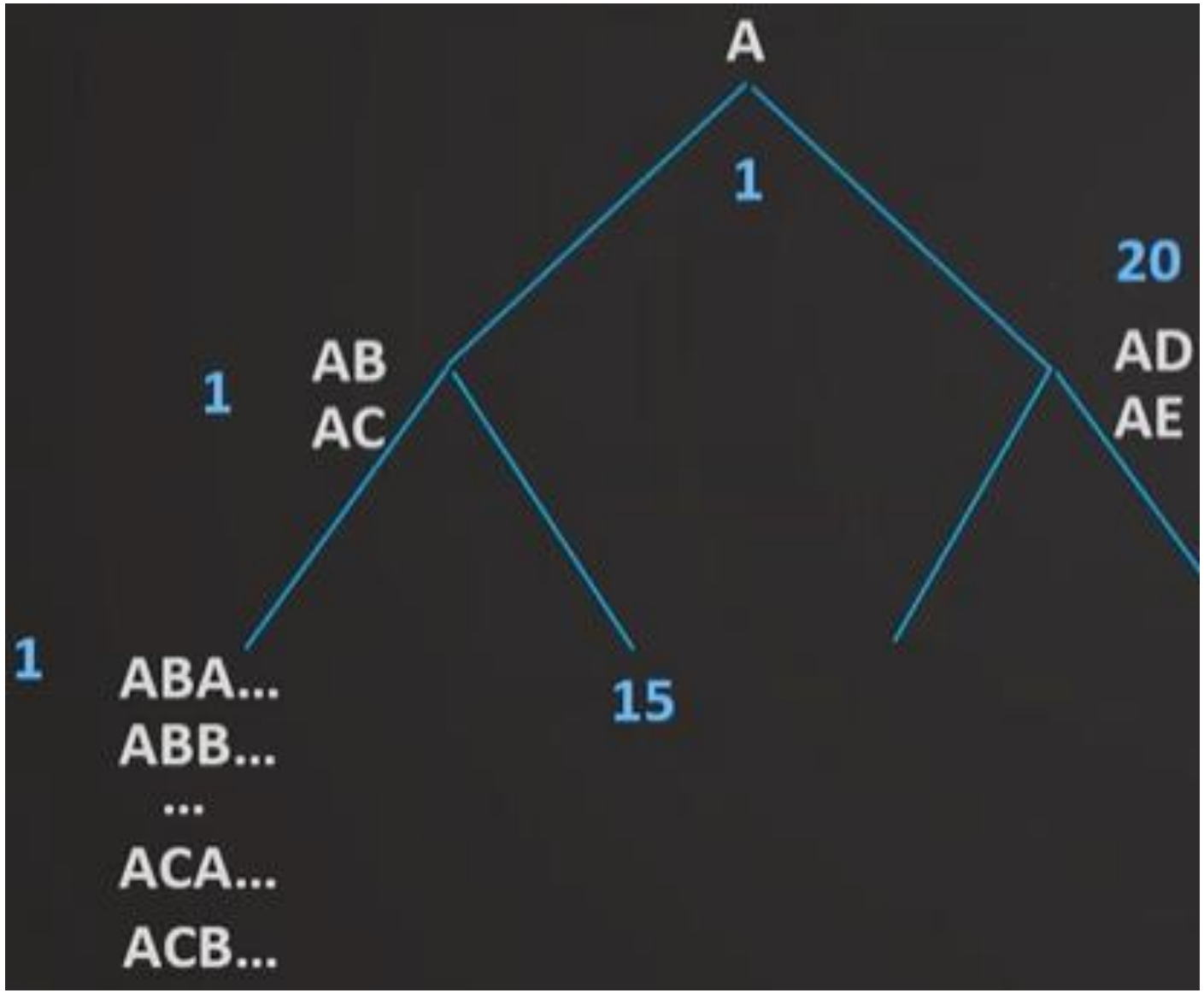
- B-Tree Indexes
- Bitmap Indexes

B-Tree Indexes

- Standard/Default Index
- Multi-level tree structure
- Breaks data into pages or blocks
- Should be used for high-cardinality

  Columns (columns with many distinct

  values)
- Not entire table (Costly)



Bitmap Indexes

- Large amount of data + low cardinality (limited number of distinct values)
- Very storage efficient
- More optimized for read
- Good for many repeating values

Tran_id	Product_id	Product_Type
1	P001	Toy
2	P002	Toy
3	P003	Toy
4	P004	Beverage
5	P005	Beverage

Row_id	Value
1	Toy
4	Beverage

Bit
11100
00011

Guidelines for Indexes

01

Identify Query Patterns

02

Prioritize Columns for Indexing

03

Evaluate Index Types

04

Regularly analyze Index Usage

05

Avoid Over indexing

06

Test and Iterate

Setting Indexes

```sql
CREATE INDEX customer_id_index ON core.sales
    (
        customer_id ASC
    );
```

```sql
CREATE INDEX index_name ON table_name [USING method]
    (
     column_name [ASC | DESC],
     ...
    );
```

Visualization and Reporting

- Identify the business needs.
- Create reports and dashboards to uncover insights.
- Popular Tools:
  - Power BI
  - Tableau
  - Qlik View/Qlik Sense
  - Google Data Studio
- Practical: Connect PostGreSQL to Power BI

Data Mining for Business Intelligence

- Uncover hidden patterns, trends and insights from large datasets
- Involves applying various statistical, machine learning, and artificial intelligence techniques to analyze data and extract valuable knowledge for decision-making.
- Use Cases:
  - Predictive Analytics
  - Customer Segmentation
  - Churn Prediction
  - Fraud Detection
  - Sentiment Analysis

Practical: Create tables in Public Schema

- Create Dimension Tables
- Create Fact Tables

Create Dimension Tables

```sql
-- Create Date_Dim Table
CREATE TABLE Date_Dim (
    date_id SERIAL PRIMARY KEY,
    date DATE,
    day INT,
    month INT,
    year INT
);

-- Create Product_Dim Table
CREATE TABLE Product_Dim (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50)
);

-- Create Customer_Dim Table
CREATE TABLE Customer_Dim (
    customer_id SERIAL PRIMARY KEY,
    customer_name VARCHAR(100),
    city VARCHAR(100),
    country VARCHAR(100)
);
```

Create Fact Table

```sql
-- Create Sales_Fact Table
CREATE TABLE Sales_Fact (
    sales_id INT PRIMARY KEY,
    date_id INT REFERENCES Date_Dim(date_id),
    product_id INT REFERENCES Product_Dim(product_id),
    customer_id INT REFERENCES Customer_Dim(customer_id),
    quantity_sold INT,
    total_amount NUMERIC(10, 2)
);
```

Insert Data in Dimension Table

```sql
-- Populate Date_Dim Table
INSERT INTO Date_Dim (date, day, month, year)
SELECT DISTINCT date_sales, EXTRACT(day FROM date_sales), EXTRACT(month FROM date_sales), EXTRACT(year FROM date_sales)
FROM "Staging"."temp_tblproduct";

-- Populate Product_Dim Table
INSERT INTO Product_Dim (product_name, category)
SELECT DISTINCT product_name, category
FROM "Staging"."temp_tblproduct";

-- Populate Customer_Dim Table
INSERT INTO Customer_Dim (customer_name, city, country)
SELECT DISTINCT customer_name, city, country
FROM "Staging"."temp_tblproduct";
```

Verify if data exists in Fact Table

```
1  SELECT * FROM Sales_Fact
```

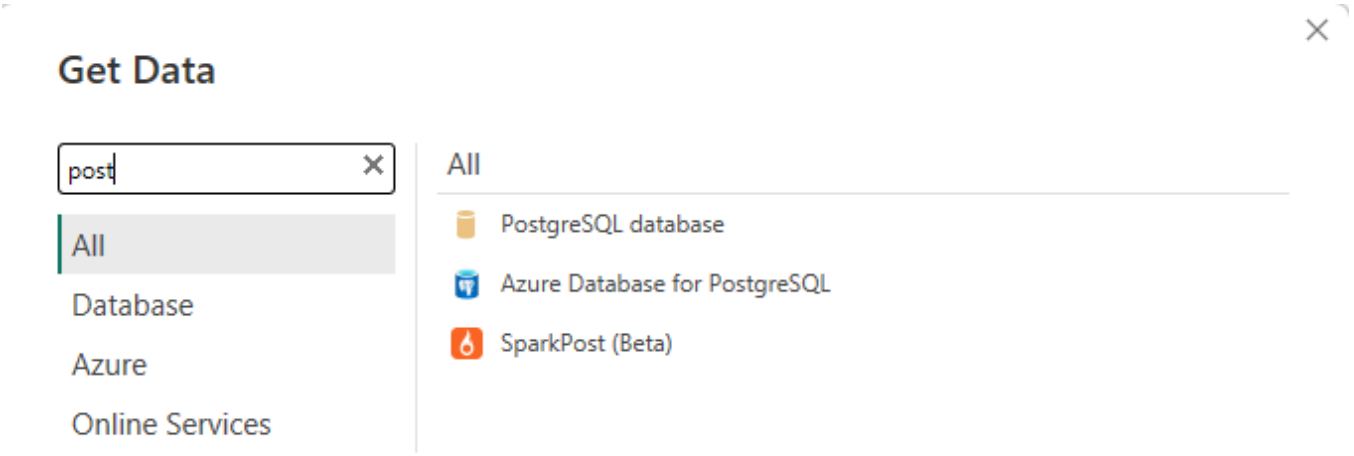
Data Output Messages Notifications

	sales_id [PK] integer	date_id integer	product_id integer	customer_id integer	price numeric (10,2)
1	1	1	4	19	1200.00
2	2	2	9	4	500.00
3	3	3	10	1	0.00
4	4	4	4	17	1100.00
5	5	5	11	5	40.00
6	6	6	1	10	0.00
7	7	7	8	14	30.00
8	8	8	12	13	300.00

- Count should match with the total data in the source CSV file.

Connect Power BI to PostGreSQL

Select PostGreSQL database from the connections

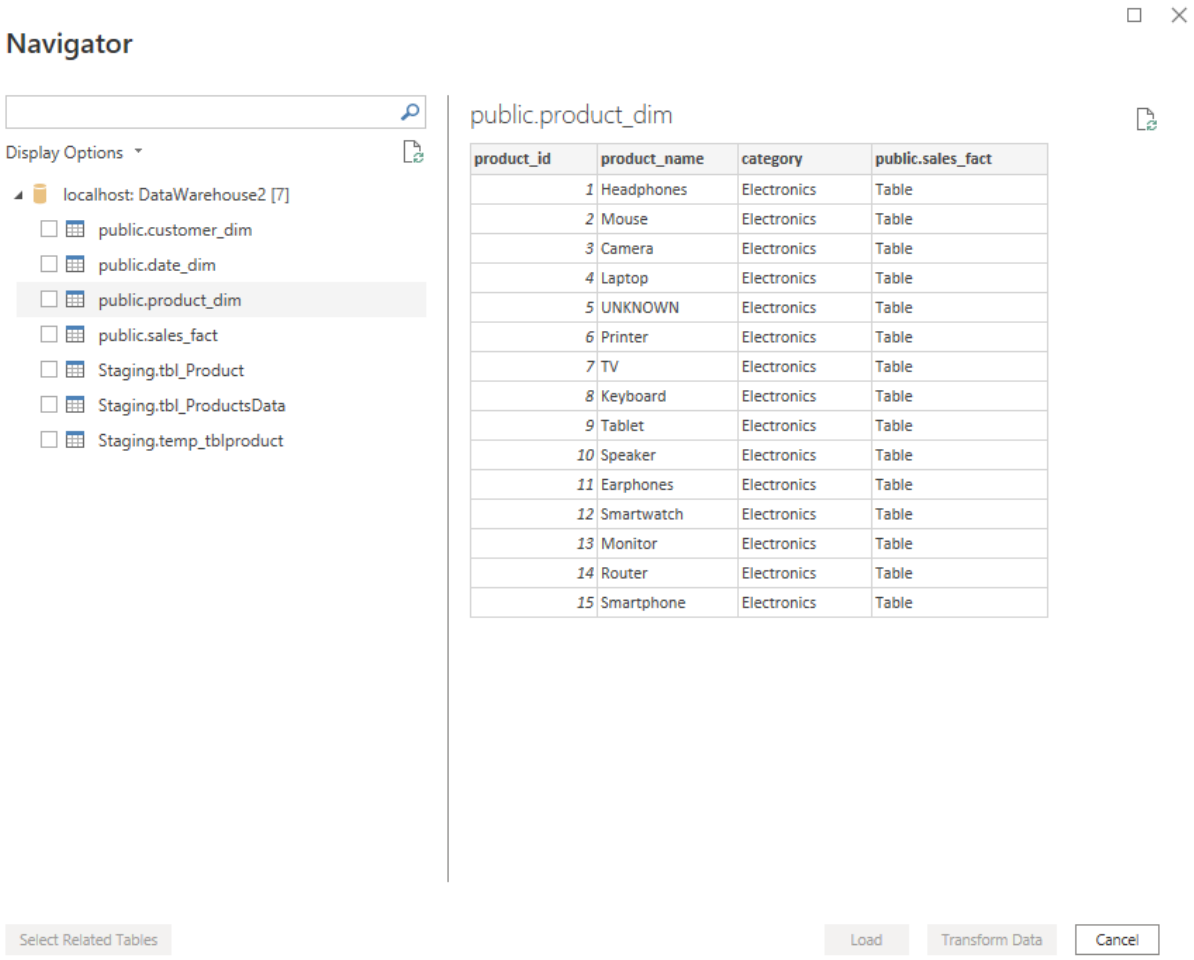


Connect Power BI to PostGreSQL

- Add the server and Database name as shown below.

Connect Power BI to PostGreSQL

- Add the username and password to get the following view.



product_id	product_name	category	public.sales_fact
1	Headphones	Electronics	Table
2	Mouse	Electronics	Table
3	Camera	Electronics	Table
4	Laptop	Electronics	Table
5	UNKNOWN	Electronics	Table
6	Printer	Electronics	Table
7	TV	Electronics	Table
8	Keyboard	Electronics	Table
9	Tablet	Electronics	Table
10	Speaker	Electronics	Table
11	Earphones	Electronics	Table
12	Smartwatch	Electronics	Table
13	Monitor	Electronics	Table
14	Router	Electronics	Table
15	Smartphone	Electronics	Table


Thank you