UNIVERSIDAD POLITECNICA SALESIANA

GESTIÓN DE BASE DE DATOS

Tema: DATAWAREHOUSE OLAP

Estudiantes:

- Juan Malo
- Jonnathan Saquicela

Docente:

Ing. Germán Parra

Esquema de datos de la solución OLAP

El modelo que se utilizó fue el estrella debido a su facilidad de comprensión y navegación, ya que este modelo organiza los datos en una tabla central de hechos, en nuestro caso dicha tabla se llama th_productos la cuál está conectada a múltiples tablas dimensionales, como dim_tiempo, dim_clientes, dim_proveedores, dim_ubicaciones, etc. De esta forma logramos facilitar la consulta para el análisis OLAP.

La estructura de estrella disminuye la complejidad de las uniones, lo cual aumenta el desempeño en consultas frecuentes como agregaciones y cálculos. Además de tener un diseño eficiente debido a que las dimensiones están desnormalizadas, lo que reduce la cantidad de tablas necesarias para responder preguntas de negocio comunes.

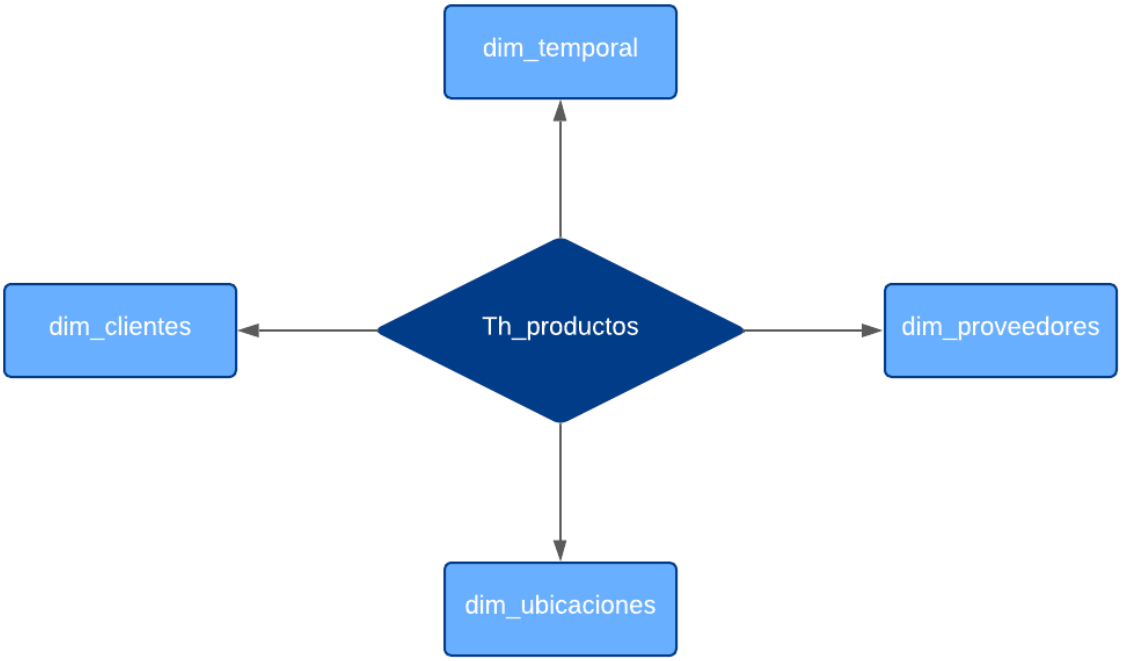
Esquema general

Tabla de hechos: th_productos

Registra datos transaccionales como ventas por producto, cliente, ubicación y tiempo.

Dimensiones principales:

- **dim_tiempo**: Permite analizar ventas en función de fechas específicas.
- **dim_clientes**: Proporciona datos detallados de los clientes.
- **dim_proveedores**: Contiene información sobre proveedores.
- **dim_ubicaciones**: Detalla ubicaciones jerárquicas (provincias, regiones, países).



Elección del aplicativo OLAP

Conexión con el proceso ETL

El código realizado demuestra un enfoque bien estructurado para el proceso ETL, donde los datos son extraídos de tablas transaccionales, transformados y cargados en estructuras diseñadas específicamente para análisis dimensiones y tabla de hechos. En este caso, el aplicativo OLAP implementado es un sistema propio, desarrollado dentro del entorno de la base de datos, que sigue principios de diseño OLAP sin depender de herramientas comerciales como Oracle OLAP. Este OLAP casero funciona eficientemente al cumplir con los siguientes requisitos:

a. **Procesamiento de datos SQL puro:** El uso extensivo de cursores en PL/SQL para realizar transformaciones de datos muestra que el aplicativo OLAP debía funcionar eficientemente con Oracle DBMS y soportar operaciones avanzadas de SQL.

b. **Capacidad para trabajar con modelos de datos dimensionales:** Los cubos OLAP construidos dependen de dimensiones como DIM_clientes, DIM_proveedores y la tabla de hechos TH_productos, reflejando un modelo estrella diseñado específicamente para consultas analíticas.

Justificación de la elección del aplicativo

El uso de un OLAP casero desarrollado directamente dentro del entorno SQL/PL-SQL es coherente con las siguientes necesidades identificadas en el código:

a. **Compatibilidad con el modelo estrella:** El OLAP casero aprovecha el diseño de un modelo estrella compuesto por dimensiones (DIM_clientes, DIM_proveedores, DIM_temporal, etc.) y una tabla de hechos (TH_productos). Este modelo organiza los datos de manera eficiente para consultas analíticas sin necesidad de herramientas externas como Oracle OLAP.

b. **Soporte nativo para transformaciones avanzadas:** Las transformaciones realizadas en el proceso ETL, como la creación de una dimensión temporal basada en fechas transaccionales o la agregación de cantidades en TH_productos, están integradas directamente en el código PL/SQL. Este enfoque permite cargar y organizar datos precalculados en una estructura optimizada para análisis multidimensionales.

c. **Flexibilidad en el desarrollo y uso:** Al implementar un OLAP casero, se obtiene control total sobre la arquitectura, lo que permite adaptarse rápidamente a cambios en los requisitos de análisis o estructura de los datos.

d. **Eliminación de dependencia de herramientas externas:** Este enfoque evita la necesidad de adquirir y configurar soluciones comerciales, reduciendo costos y complejidad operativa, mientras que mantiene un rendimiento adecuado para las necesidades del negocio.

Ventajas del OLAP casero

a) **Adaptabilidad al entorno actual:** Este OLAP casero está diseñado específicamente para trabajar en el entorno DBMS existente, maximizando el uso de sus capacidades nativas sin requerir extensiones o módulos adicionales.
b) **Procesamiento eficiente de datos:** El uso de PL/SQL y SQL puro asegura que las transformaciones y cargas sean manejadas de manera eficiente dentro del mismo sistema.
c) **Escalabilidad:** Aunque el OLAP casero es más simple que una solución comercial, el modelo estrella permite agregar nuevas dimensiones, métricas y optimizaciones según sea necesario.

d) **Consultas personalizadas:** Las consultas analíticas pueden ser diseñadas directamente sobre las tablas dimensionales y de hechos, logrando análisis complejos como rankings, agregaciones y tendencias sin depender de un motor OLAP externo.

Configuración del DBMS y OLAP

La configuración del DBMS y las tablas dimensionales en esta solución refleja un modelo estrella bien diseñado, adaptado para el uso de un OLAP casero. Este modelo organiza los datos de manera eficiente para análisis multidimensionales, donde las dimensiones alimentan una tabla de hechos central.

Configuración de las tablas dimensionales

Las tablas dimensionales (DIM) son clave para soportar las consultas OLAP al proporcionar datos categóricos con niveles jerárquicos y atributos descriptivos. Estas tablas fueron configuradas de la siguiente manera

a. DIM_tiempo

Propósito: Permite analizar los datos de ventas por periodos de tiempo como trimestrales, anuales, etc.

Configuración:

```
CREATE TABLE dim_tiempo(
      fechaid int,
      fechaorden date,
      CONSTRAINT PK_dim_tiempo PRIMARY KEY (fechaid)
);
```

Población de datos: Mediante un cursor que recorre las fechas únicas en las órdenes

```
DECLARE
      CURSOR fechas_cur IS SELECT DISTINCT(fechaorden) FROM
ordenes;
      pointer INT := 1;
      f_value DATE;
 BEGIN
      OPEN fechas_cur;
      LOOP
            FETCH fechas_cur INTO f_value;
            EXIT WHEN fechas_cur%NOTFOUND;
            INSERT INTO dim_tiempo VALUES (pointer, f_value);
```

```
            pointer := pointer + 1;
        END LOOP;
    CLOSE fechas_cur;
    END;
```

Aporte al análisis: Facilita consultas como ventas por año, mes o día.

b. DIM_proveedores

Propósito: Permite categorizar y analizar las ventas por proveedor.

Configuración:

```
CREATE TABLE DIM_proveedores (
    proveedorid INT,
    nombreprov VARCHAR2(50),
    CONSTRAINT PK_DIM_PROVEEDORES PRIMARY KEY (proveedorid)
);
```

Población de datos: Se realiza copiando información desde la tabla **PROVEEDORES:**

```
DECLARE
    CURSOR prov_cur IS SELECT proveedorid, nombreprov FROM
proveedores;
    p_id INT;
    n_prov VARCHAR2(50);
BEGIN
    OPEN prov_cur;
    LOOP
        FETCH prov_cur INTO p_id, n_prov;
        EXIT WHEN prov_cur%NOTFOUND;
        INSERT INTO DIM_proveedores VALUES (p_id, n_prov);
    END LOOP;
    CLOSE prov_cur;
END;
```

Aporte al análisis: Permite identificar el rendimiento de proveedores y analizar la procedencia de productos.

c. DIM_clientes

Propósito: Permite analizar ventas y comportamiento de los clientes.

Configuración:

```
CREATE TABLE DIM_clientes (
    clienteid INT,
    nombrecontacto VARCHAR2(50),
    CONSTRAINT PK_DIM_CLIENTES PRIMARY KEY (clienteid)
);
```

Población de datos: Usando información de la tabla **CLIENTES:**

```
DECLARE
    CURSOR cli_cur IS SELECT clienteid, nombrecontacto FROM
clientes;
    cli_id INT;
    cli_n VARCHAR2(50);
BEGIN
    OPEN cli_cur;
    LOOP
        FETCH cli_cur INTO cli_id, cli_n;
        EXIT WHEN cli_cur%NOTFOUND;
        INSERT INTO DIM_clientes VALUES (cli_id, cli_n);
    END LOOP;
    CLOSE cli_cur;
END;
```

Aporte al análisis: Ayuda a identificar tendencias y patrones de compra.

d. DIM_ubicaciones

Propósito: Permite el análisis geográfico de las ventas por regiones, países y provincias.

Configuración:

```
CREATE TABLE DIM_ubicaciones (
    provinciaid INT,
    nombreprovincia VARCHAR2(50),
    paisid INT,
    regionid INT,
    CONSTRAINT PK_DIM_PROVINCIAS PRIMARY KEY (provinciaid),
    CONSTRAINT FK_SDIM_REGIONES_UBIC FOREIGN KEY (regionid)
REFERENCES SDIM_regiones (regionid),
    CONSTRAINT FK_SDIM_PAISES_UBIC FOREIGN KEY (paisid) REFERENCES
SDIM_paises (paisid)
```

```
);
```

Población de datos: Usando cursores para extraer información jerárquica de **PROVINCIAS**, **PAISES** y **REGIONES**:

```
DECLARE
    CURSOR prov_cur IS SELECT * FROM provincias;
    prov_id INT;
    prov_nom VARCHAR2(50);
    r_id INT;
    p_id INT;
BEGIN
    OPEN prov_cur;
    LOOP
        FETCH prov_cur INTO prov_id, prov_nom, r_id, p_id;
        EXIT WHEN prov_cur%NOTFOUND;
        INSERT INTO DIM_ubicaciones VALUES (prov_id, prov_nom, p_id,
r_id);
    END LOOP;
    CLOSE prov_cur;
END;
```

Aporte al análisis: Proporciona visibilidad sobre el rendimiento en ubicaciones específicas.

Proceso de ETL

El proceso de carga para esta dimensión sigue el patrón de extracción-transformación-carga (ETL). Se detalla a continuación:

Extracción

Los datos son extraídos directamente de las tablas de la base de datos mediante un cursor SQL que selecciona los campos, en este caso es **clienteid** y **nombrecontacto**:

```
DECLARE
    CURSOR cli_cur IS
        SELECT clienteid, nombrecontacto FROM clientes;
```

Transformación

En este caso, no se realizan transformaciones complejas sobre los datos. Se asegura que cada cliente tenga una entrada única en la tabla dimensional.

Carga

Los datos extraídos son insertados directamente en **DIM_clientes.** El proceso utiliza un bucle para recorrer el cursor y cargar cada registro en la dimensión

```
BEGIN
    OPEN cli_cur;
    LOOP
        FETCH cli_cur INTO cli_id, cli_n;
        EXIT WHEN cli_cur%NOTFOUND;
        INSERT INTO dim_clientes VALUES (cli_id, cli_n);
    END LOOP;
    CLOSE cli_cur;
END;
```

Validación

Tras la carga, se realiza una consulta para confirmar que los datos han sido correctamente insertados:

```
SELECT * FROM dim_clientes;
```

Esto se repite para las demás dimensiones