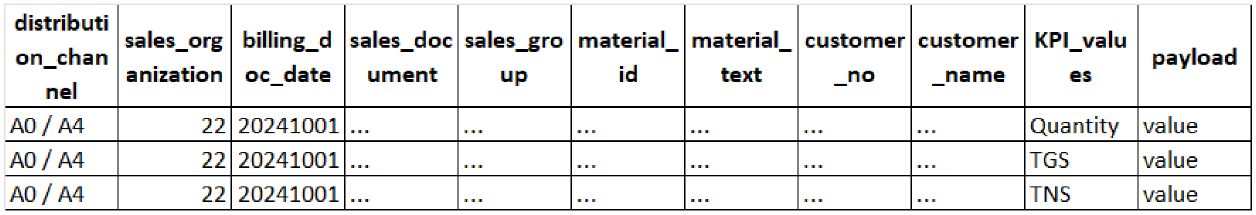
Data Vault Modeling:

To construct a Data Vault structure, I begin by identifying and populating the Hubs, Links, and Satellites in accordance with Data Vault modeling principles.

Although no specific source system or tables have been provided, I approach this by analyzing the output of the Common Table Expression (CTE). From this analysis, I derive the necessary components—Hubs, Links, and Satellites—based on the patterns, keys, and business rules inferred from the CTE's results

Assuming the below as the output format of the CTE:



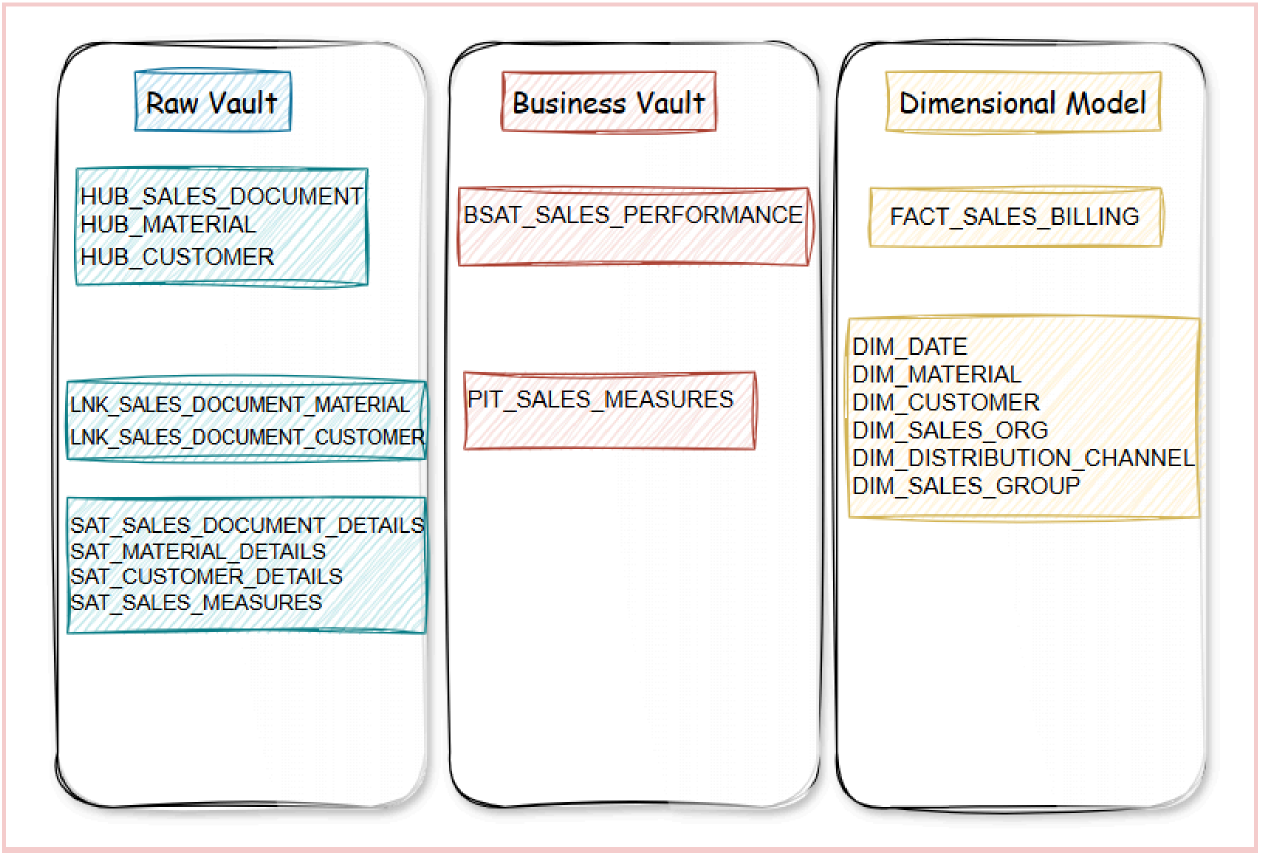
distribution_channel	sales_organization	billing_doc_date	sales_document	sales_group	material_id	material_text	customer_no	customer_name	KPI_values	payload
A0 / A4	22	20241001	...	...	...	...	...	...	Quantity	value
A0 / A4	22	20241001	...	...	...	...	...	...	TGS	value
A0 / A4	22	20241001	...	...	...	...	...	...	TNS	value

Analyzing the Source Data (from the CTE)

Based on the CTE, the key elements and potential business keys are:

- **sales_document (c.kaufn): Seems like a core business key for a sales transaction.**
- **material_id (c.artnr): A business key for a product or service.**
- **customer_no (k.kunnr): A business key for a customer.**
- **billing_doc_date (c.budat): A date associated with the billing document.**
- **distribution_channel (c.vtweg): An attribute describing the sales process.**
- **sales_organization (c.vkorg): An attribute describing the sales organizational unit.**
- **sales_group (c.vkgrp): An attribute describing the sales group.**
- **material_text (m.maktx): An attribute describing the material.**
- **customer_name (k.nam1): An attribute describing the customer.**
- **Quantity, TGS, TNS: Measures related to the sales/billing event**

Data Vault to Dimensional Flow (End - to - End):



1.RAW Vault - "Source Data with full history"

Purpose:

The unaltered source data is captured in Raw Vault with a focus on traceability, auditability and historical tracking.

The data is structured into:

Hubs: Unique business keys (e.g., Customers, Materials, Sales Documents)
Links: Relationships between those keys
Satellites: Descriptive and historical attributes or metrics

And the standard Data Vault metadata columns are also included as follows:

- hk (Hash Key) - A surrogate key, typically a hash of the business key(s).
- load_dts (Load Date Timestamp) - The timestamp when the record was loaded into the vault.
- rec_src (Record Source) - The source system or file from which the record originated.

- hashdiff (Hash Difference) - A hash of the descriptive attributes in a Satellite, used to detect changes.

| Raw Vault Components|

Hub

1. HUB_SALES_DOCUMENT
2. HUB_MATERIAL
3. HUB_CUSTOMER

Reason for Choice: Hub tables are the core identifiers for business entities (e.g., sales documents, materials, and customers).

Link

1. LNK_SALES_DOCUMENT_MATERIAL
2. LNK_SALES_DOCUMENT_CUSTOMER

Reason for Choice:

- Link tables establish relationships between business entities. In this case, Sales Documents are linked to Materials and Customers.
- LNK_SALES_DOCUMENT_MATERIAL links the Sales Document with the Material (many-to-many relationship).
- LNK_SALES_DOCUMENT_CUSTOMER links the Sales Document with the Customer.
- These links ensure that relationships are tracked efficiently without redundancy and are key to handling many-to-many associations.

Satellite

1. SAT_SALES_DOCUMENT_DETAILS
2. SAT_MATERIAL_DETAILS
3. SAT_CUSTOMER_DETAILS
4. SAT_SALES_MEASURES

Reason for Choice:

- Satellite tables store historical and descriptive data for the entities in the Hub.Satellites store data that changes over time, ensuring that historical context is preserved for reporting and analytics.

    - SAT_SALES_DOCUMENT_DETAILS stores additional attributes of sales documents like distribution channel and sales group.
    - SAT_MATERIAL_DETAILS stores descriptive information about the material, like the text description.
    - SAT_CUSTOMER_DETAILS tracks changes in customer details, like the customer's name.
    - SAT_SALES_MEASURES tracks metrics such as quantity, TGS, and TNS for each sales document-material combination.

SQL code for the creation of Hubs, Links, and Satellite :

-- HUBS: Unique Business Keys

```
CREATE TABLE raw_vault.HUB_CUSTOMER (

    hk_customer BINARY(16) PRIMARY KEY,

    customer_no_bk VARCHAR(50) UNIQUE NOT NULL,

    load_dts TIMESTAMP NOT NULL,

    rec_src VARCHAR(255) NOT NULL

);


CREATE TABLE raw_vault.HUB_MATERIAL (

    hk_material BINARY(16) PRIMARY KEY,

    material_id_bk VARCHAR(50) UNIQUE NOT NULL,

    load_dts TIMESTAMP NOT NULL,

    rec_src VARCHAR(255) NOT NULL

);


CREATE TABLE raw_vault.HUB_SALES_DOCUMENT (
    hk_sales_document BINARY(16) PRIMARY KEY,
    sales_document_bk VARCHAR(50) UNIQUE NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    rec_src VARCHAR(255) NOT NULL
);
```

-- LINKS: Relationships Between Keys

```
CREATE TABLE raw_vault.LNK_SALES_DOCUMENT_CUSTOMER (
    hk_sales_document_customer BINARY(16) PRIMARY KEY,
    hk_sales_document BINARY(16) NOT NULL,
    hk_customer BINARY(16) NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    rec_src VARCHAR(255) NOT NULL
);

CREATE TABLE raw_vault.LNK_SALES_DOCUMENT_MATERIAL (
    hk_sales_document_material BINARY(16) PRIMARY KEY,
    hk_sales_document BINARY(16) NOT NULL,
    hk_material BINARY(16) NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    rec_src VARCHAR(255) NOT NULL
);
```

-- SATELLITES: Descriptive & Historical Attributes

```
CREATE TABLE raw_vault.SAT_CUSTOMER_DETAILS (
    hk_customer BINARY(16) NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    hashdiff BINARY(16) NOT NULL,
    rec_src VARCHAR(255) NOT NULL,
    customer_name VARCHAR(255),
    PRIMARY KEY (hk_customer, load_dts)
);

CREATE TABLE raw_vault.SAT_MATERIAL_DETAILS (
    hk_material BINARY(16) NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    hashdiff BINARY(16) NOT NULL,
    rec_src VARCHAR(255) NOT NULL,
    material_text VARCHAR(255),
    PRIMARY KEY (hk_material, load_dts)
);

CREATE TABLE raw_vault.SAT_SALES_DOCUMENT_DETAILS (
    hk_sales_document BINARY(16) NOT NULL,
    load_dts TIMESTAMP NOT NULL,
    hashdiff BINARY(16) NOT NULL,
    rec_src VARCHAR(255) NOT NULL,
    billing_doc_date DATE,
    distribution_channel VARCHAR(10),
    sales_organization VARCHAR(10),
    sales_group VARCHAR(10),
    PRIMARY KEY (hk_sales_document, load_dts)
```

```
);


CREATE TABLE raw_vault.SAT_SALES_MEASURES (

    hk_sales_document_material BINARY(16) NOT NULL,

    load_dts TIMESTAMP NOT NULL,

    hashdiff BINARY(16) NOT NULL,

    rec_src VARCHAR(255) NOT NULL,

    quantity DECIMAL(18, 3),

    tgs DECIMAL(18, 3),

    tns DECIMAL(18, 3),

    PRIMARY KEY (hk_sales_document_material, load_dts)

);
```

2. BUSINESS VAULT – “Applying Business Logic”

| Business Vault Components |

Business Satellite (BSAT)

1. BSAT_SALES_PERFORMANCE

Reason for Choice:

- This table stores calculated or aggregated data based on business logic or performance metrics. It's a business-specific calculation that can combine data from raw satellites (like SAT_SALES_MEASURES) to produce a consolidated, ready-to-use metric.

- BSAT_SALES_PERFORMANCE stores metrics like quantity, TGS, and TNS, but in a more business-friendly format ready for reporting. Although these measures are already calculated in your source CTE, in a typical ETL process loading a Data Vault, these calculations would happen during the loading of the Business Vault from the Raw Vault.

Point-in-Time (PIT) Table

1. PIT_SALES_MEASURES

Reason for Choice:

- A Point-in-Time (PIT) table helps with querying the latest or a specific point-in-time version of one or more Satellites without complex joins. This PIT table would allow efficient querying of the `BSAT_SALES_PERFORMANCE` and potentially other relevant Satellites (like `SAT_SALES_DOCUMENT_DETAILS`, `SAT_MATERIAL_DETAILS`, `SAT_CUSTOMER_DETAILS`) at a given date.

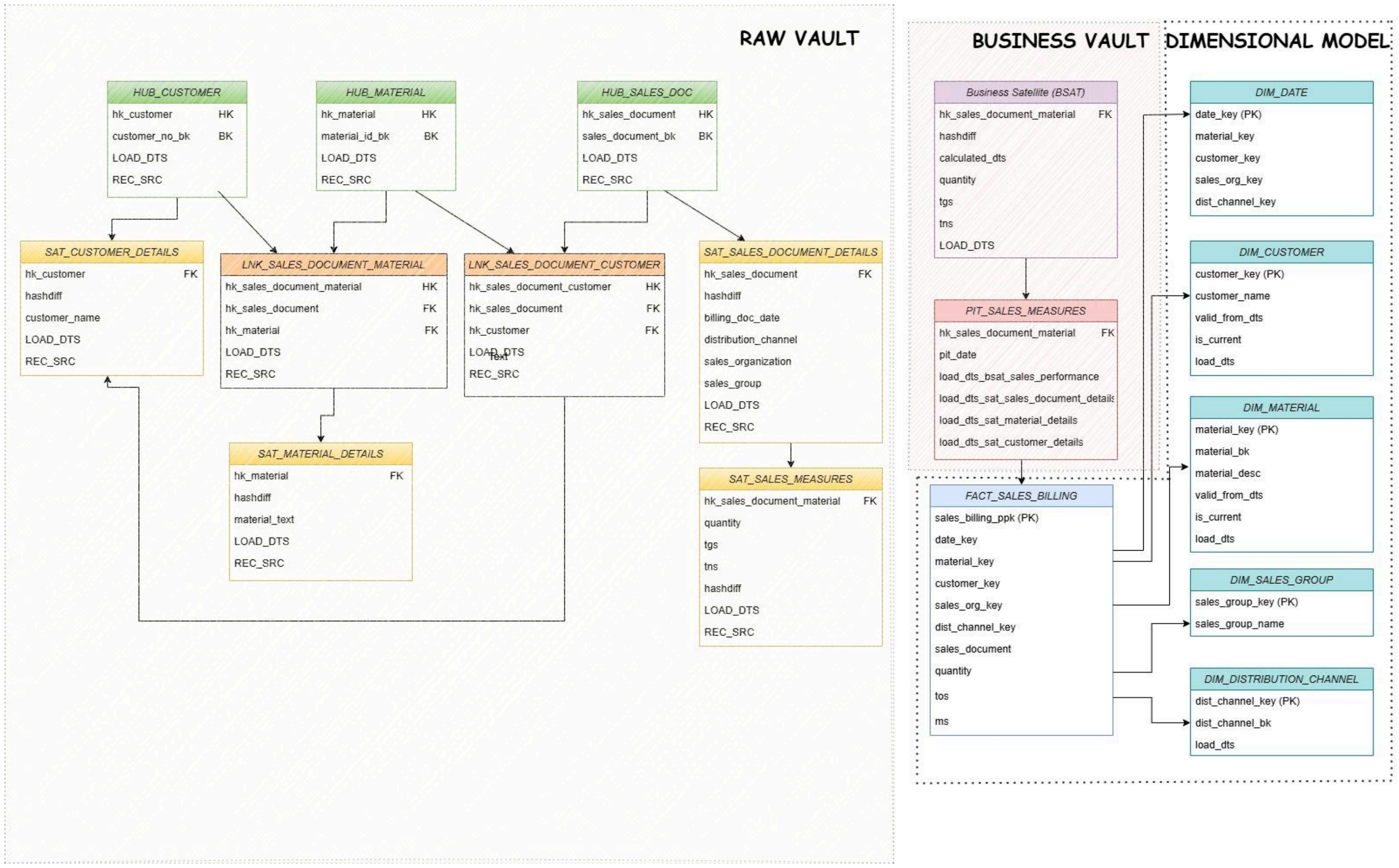
-- Business Satellite: Derived metrics or calculations

```
CREATE TABLE business_vault.BSAT_SALES_PERFORMANCE (

    hk_sales_document_material BINARY(16) NOT NULL,

    load_dts TIMESTAMP NOT NULL,

    hashdiff BINARY(16) NOT NULL,

    calculated_dts TIMESTAMP NOT NULL,

    quantity DECIMAL(18, 3),

    tgs DECIMAL(18, 3),

    tns DECIMAL(18, 3),

    PRIMARY KEY (hk_sales_document_material, load_dts)

);
```

-- PIT Table: Snapshot of the latest data at a point in time

```
CREATE TABLE business_vault.PIT_SALES_MEASURES (

    pit_date DATE NOT NULL,

    hk_sales_document_material BINARY(16) NOT NULL,
    load_dts_bsat_sales_performance TIMESTAMP,

    load_dts_sat_sales_document_details TIMESTAMP,

    load_dts_sat_material_details TIMESTAMP,

    load_dts_sat_customer_details TIMESTAMP,

    PRIMARY KEY (pit_date, hk_sales_document_material)

);
```

3. DIMENSIONAL MODEL – “Optimized for Reporting”



Purpose:This layer transforms the business logic into easy-to-query star schemas used for dashboards and analytics. It consists of:

1. Fact Tables here store measures and numeric data.
2. Dimension Tables provide context (who, what, when, where).

Fact Table

1. FACT_SALES_BILLING

Reason for Choice:

- The Fact table stores the quantitative data related to the sales transactions, including metrics like quantity, TGS, and TNS.
- This fact table is linked to the dimensions (e.g., DIM_DATE, DIM_MATERIAL, DIM_CUSTOMER) for easy aggregation and reporting.
- The FACT_SALES_BILLING uses surrogate keys (from the dimension tables) to link facts with their corresponding dimensional data for analysis.

Dimension Tables

1. DIM_DATE
2. DIM_MATERIAL
3. DIM_CUSTOMER
4. DIM_SALES_ORG
5. DIM_DISTRIBUTION_CHANNEL
6. DIM_SALES_GROUP

Reason for Choice:

- Dimension tables are used in reporting and analytics to provide context to the facts in the fact table (FACT_SALES_BILLING).
- The DIM_MATERIAL and DIM_CUSTOMER use SCD Type 2 to track changes in descriptions and names over time (i.e., maintaining a historical view).

- Other dimensions like DIM_SALES_ORG, DIM_DISTRIBUTION_CHANNEL, and DIM_SALES_GROUP are designed with SCD Type 1 or static attributes because they don't change frequently and don't require historical tracking.

-- DIMENSIONS

```
CREATE TABLE reporting_layer.DIM_DATE (

    date_key INT PRIMARY KEY, -- Format: YYYYMMDD

    full_date DATE UNIQUE NOT NULL,

    calendar_year INT,

    calendar_month INT,

    day_of_month INT,

    day_of_week INT,

    day_name VARCHAR(10),

    month_name VARCHAR(10)

);


CREATE TABLE reporting_layer.DIM_CUSTOMER (

    customer_key INT IDENTITY(1,1) PRIMARY KEY,

    customer_no_bk VARCHAR(50),

    customer_name VARCHAR(255),

    valid_from_dts TIMESTAMP,

    valid_to_dts TIMESTAMP,

    is_current BOOLEAN,

    load_dts TIMESTAMP

);
```

```
CREATE TABLE reporting_layer.DIM_MATERIAL (
    material_key INT IDENTITY(1,1) PRIMARY KEY,
    material_id_bk VARCHAR(50),
    material_text VARCHAR(255),
    valid_from_dts TIMESTAMP,
    valid_to_dts TIMESTAMP,
    is_current BOOLEAN,
    load_dts TIMESTAMP
);

CREATE TABLE reporting_layer.DIM_SALES_ORG (
    sales_org_key INT IDENTITY(1,1) PRIMARY KEY,
    sales_organization_bk VARCHAR(10),
    sales_organization_name VARCHAR(100),
    load_dts TIMESTAMP
);

CREATE TABLE reporting_layer.DIM_DISTRIBUTION_CHANNEL (
    dist_channel_key INT IDENTITY(1,1) PRIMARY KEY,
    distribution_channel_bk VARCHAR(10),
    distribution_channel_name VARCHAR(100),
    load_dts TIMESTAMP
);

CREATE TABLE reporting_layer.DIM_SALES_GROUP (
    sales_group_key INT IDENTITY(1,1) PRIMARY KEY,
    sales_group_bk VARCHAR(10),

    sales_group_name VARCHAR(100),

    load_dts TIMESTAMP

);
```

-- FACT: Numerical Sales Data

```
CREATE TABLE reporting_layer.FACT_SALES_BILLING (

    sales_billing_pk INT IDENTITY(1,1) PRIMARY KEY,

    date_key INT NOT NULL,

    material_key INT NOT NULL,

    customer_key INT NOT NULL,

    sales_org_key INT NOT NULL,

    dist_channel_key INT NOT NULL,

    sales_group_key INT NOT NULL,

    sales_document VARCHAR(50),

    quantity DECIMAL(18, 3),

    tgs DECIMAL(18, 3),

    tns DECIMAL(18, 3),

    load_dts TIMESTAMP,


    FOREIGN KEY (date_key) REFERENCES reporting_layer.DIM_DATE(date_key),

    FOREIGN KEY (material_key) REFERENCES
reporting_layer.DIM_MATERIAL(material_key),

    FOREIGN KEY (customer_key) REFERENCES
reporting_layer.DIM_CUSTOMER(customer_key),

    FOREIGN KEY (sales_org_key) REFERENCES
reporting_layer.DIM_SALES_ORG(sales_org_key),
    FOREIGN KEY (dist_channel_key) REFERENCES
reporting_layer.DIM_DISTRIBUTION_CHANNEL(dist_channel_key),

    FOREIGN KEY (sales_group_key) REFERENCES
reporting_layer.DIM_SALES_GROUP(sales_group_key)

);
```

In the end, we took the logic from the CTE—basically a mix of sales, customer, and product data with a few calculated KPIs—and broke it down into a proper Data Vault structure. We separated the core business keys into Hubs, hooked them together with Links, and added all the extra info like sales group, billing date, and KPIs into Satellites. This setup makes the data easier to manage and track over time. From there, we shaped it into a clean star schema that mirrors the original query, making it ready for reporting and analysis.