Laporan Dokumentasi Teknis

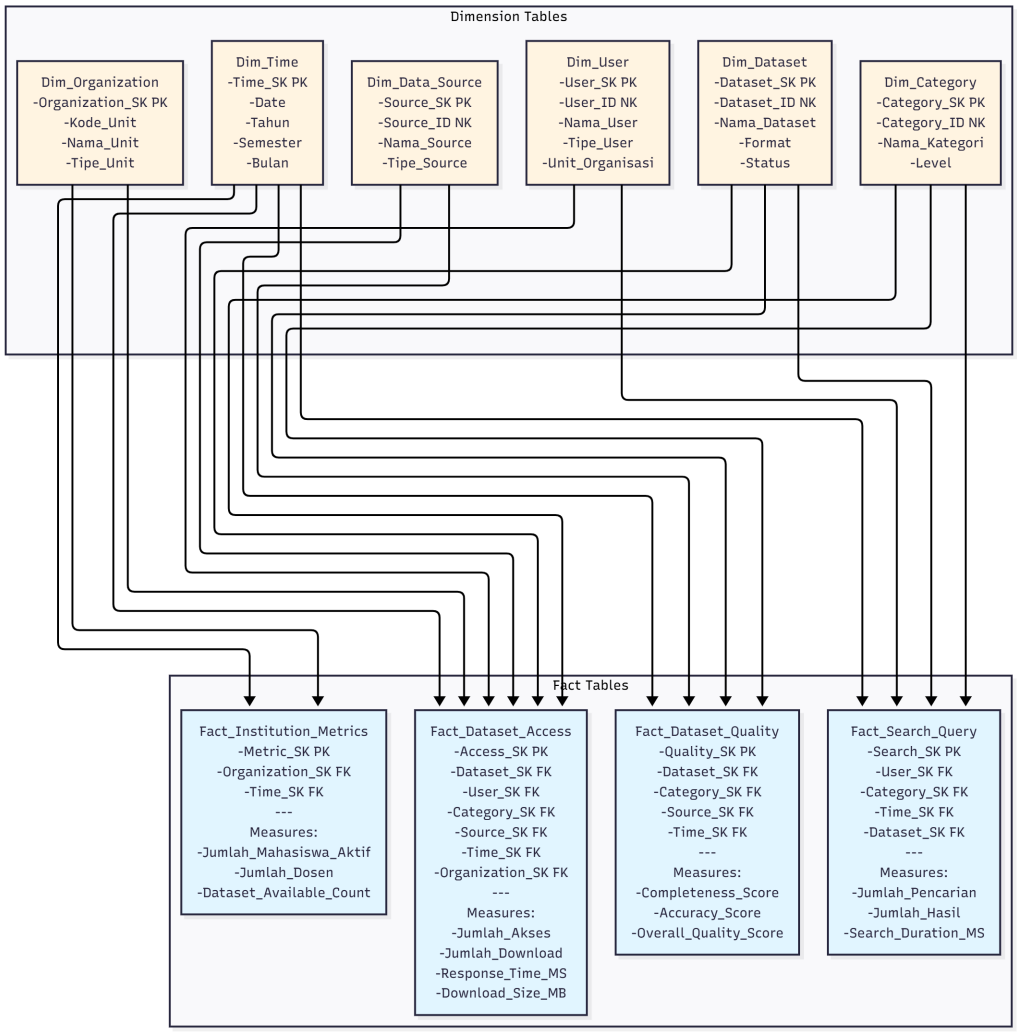
Misi 2: Desain Fisikal & Development

Kelompok 10

1 Perancangan Database

1.1 Skema Database (Star Schema)

Pada proyek Data Mart ini, kami memilih menggunakan pendekatan *Dimensional Modeling* dengan bentuk *Star Schema*. Alasan utamanya adalah karena struktur ini lebih sederhana dan optimal untuk kebutuhan analisis dibandingkan skema normalisasi biasa. Skema ini terdiri dari satu tabel fakta utama yang dikelilingi oleh beberapa tabel dimensi.Tetapi disini menggunaka 4 tabel fakta dan 6 tabel dimensi yang menjadi konteks



Gambar 1: Dimensional Model (dbo.schema) Data Mart Portal Satu Data

1.2 Implementasi Fisik (DDL)

Berikut adalah contoh implementasi kode DDL (*Data Definition Language*) untuk tabel fakta utama, yaitu `Fact_Dataset_Access`. Tabel ini berfungsi sebagai penghubung antar dimensi menggunakan *Surrogate Key* (SK).

```
CREATE TABLE dbo.Fact_Dataset_Access (
    Access_SK BIGINT IDENTITY(1,1) PRIMARY KEY,

    -- Foreign Keys
    Dataset_SK INT NOT NULL,
    User_SK INT NOT NULL,
    Category_SK INT NOT NULL,
    Organization_SK INT NOT NULL,
    Time_SK INT NOT NULL,
    Data_Source_SK INT NOT NULL,

    -- Measures
    Jumlah_Akses INT DEFAULT 1,
    Jumlah_Download INT DEFAULT 0,
    Jumlah_View INT DEFAULT 0,
    Jumlah_API_Call INT DEFAULT 0,
    File_Size_Downloaded BIGINT,
    Response_Time_MS INT,
    Success_Flag INT,

    -- Hubungan
    CONSTRAINT FK_Access_Dataset FOREIGN KEY (Dataset_SK)
    REFERENCES dbo.Dim_Dataset(Dataset_SK),
    CONSTRAINT FK_Access_User FOREIGN KEY (User_SK) REFERENCES dbo.
    Dim_User(User_SK),
    CONSTRAINT FK_Access_Category FOREIGN KEY (Category_SK)
    REFERENCES dbo.Dim_Category(Category_SK),
    CONSTRAINT FK_Access_Organization FOREIGN KEY (Organization_SK)
     REFERENCES dbo.Dim_Organization(Organization_SK),
    CONSTRAINT FK_Access_Time FOREIGN KEY (Time_SK) REFERENCES dbo.
    Dim_Time(Time_SK),
    CONSTRAINT FK_Access_DataSource FOREIGN KEY (Data_Source_SK)
    REFERENCES dbo.Dim_Data_Source(Data_Source_SK)
);
```

Listing 1: Script Pembuatan salah satu Tabel Fakta

2 Strategi Optimasi Performa

2.1 Penggunaan Indexing

Agar proses pengambilan data (*query retrieval*) tetap cepat meskipun datanya banyak, kami menerapkan dua strategi indexing:

- **Non-Clustered Index:** Kami pasang di setiap kolom *Foreign Key* pada tabel fakta. Tujuannya biar proses *JOIN* ke tabel dimensi jadi lebih ngebut.

- **Columnstore Index:** Kami pakai ini di tabel fakta. Index jenis ini sangat ampuh untuk mempercepat operasi agregasi seperti `SUM` atau `AVG` karena cara penyimpanannya berbasis kolom, bukan baris.

```
-- Index untuk Fact_Dataset_Access
CREATE NONCLUSTERED INDEX IX_Fact_Access_DatasetSK ON dbo.
    Fact_Dataset_Access(Dataset_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Access_UserSK ON dbo.
    Fact_Dataset_Access(User_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Access_TimeSK ON dbo.
    Fact_Dataset_Access(Time_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Access_CategorySK ON dbo.
    Fact_Dataset_Access(Category_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Access_OrgSK ON dbo.
    Fact_Dataset_Access(Organization_SK);

-- Index untuk Fact_Dataset_Quality
CREATE NONCLUSTERED INDEX IX_Fact_Quality_DatasetSK ON dbo.
    Fact_Dataset_Quality(Dataset_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Quality_TimeSK ON dbo.
    Fact_Dataset_Quality(Time_SK);

-- Index untuk Fact_Search_Query
CREATE NONCLUSTERED INDEX IX_Fact_Search_UserSK ON dbo.
    Fact_Search_Query(User_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Search_TimeSK ON dbo.
    Fact_Search_Query(Time_SK);

-- Index untuk Fact_Institution_Metrics
CREATE NONCLUSTERED INDEX IX_Fact_Metric_OrgSK ON dbo.
    Fact_Institution_Metrics(Organization_SK);
CREATE NONCLUSTERED INDEX IX_Fact_Metric_TimeSK ON dbo.
    Fact_Institution_Metrics(Time_SK);

-- Columnstore Index
-- Diperlukan untuk query agregasi pada tabel besar

-- Menerapkan pada tabel transaksi terbesar: Fact_Dataset_Access
    karena
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Dataset_Access
ON dbo.Fact_Dataset_Access
(
    Dataset_SK,
    User_SK,
    Time_SK,
    Jumlah_Akses,
    Jumlah_Download,
    File_Size_Downloaded
);
```

Listing 2: Script Implementasi Index

2.2 Strategi Partitioning

Untuk menangani pertumbuhan data di masa depan, kami menerapkan **Horizontal Partitioning** berdasarkan tahun (`Time_SK`). Dengan cara ini, saat kita melakukan query untuk tahun tertentu, SQL Server cukup memindai partisi tahun itu saja tanpa perlu mengecek seluruh data (teknik ini disebut *Partition Elimination*).

	Nomor Partisi	Batas Tahun (Boundary)	Jumlah Data
1	1	NULL	3080
2	2	20240101	8120
3	3	20250101	7124
4	4	20260101	0

Gambar 2: Bukti Implementasi Partisi Tahunan (Yearly)

3 Alur Proses ETL

3.1 Staging Area

Sebelum data masuk ke Data Warehouse utama, kami menampungnya dulu di *Staging Area* (Schema `stg`). Tabel-tabel di sini dibuat dengan struktur *flat* (tanpa banyak relasi/constraint) supaya proses insert data mentah bisa berjalan secepat mungkin.

3.2 Mekanisme ETL (Stored Procedures)

Seluruh proses transformasi data dilakukan secara otomatis menggunakan *Stored Procedures*. Beberapa logika penting yang kami terapkan antara lain:

- **Lookup Dimensi:** Mencari *Surrogate Key* di tabel dimensi berdasarkan ID asli dari sumber data.
- **SCD Type 2:** Untuk tabel dimensi Dataset, kami menerapkan SCD Type 2 agar bisa melacak riwayat perubahan data .
- **Data Cleaning:** Kami memfilter data transaksi yang ID-nya tidak ditemukan di tabel master agar tidak mengotori Data Warehouse.

```
-- Mengisi Tabel Fakta dengan Lookup ke Dimensi
INSERT INTO dbo.Fact_Dataset_Access (...)
SELECT
    d.Dataset_SK,
    u.User_SK,
    ...
FROM stg.Access_Log s
LEFT JOIN dbo.Dim_Dataset d ON s.Dataset_ID = d.Dataset_ID AND d.
   Is_Current = 1
```

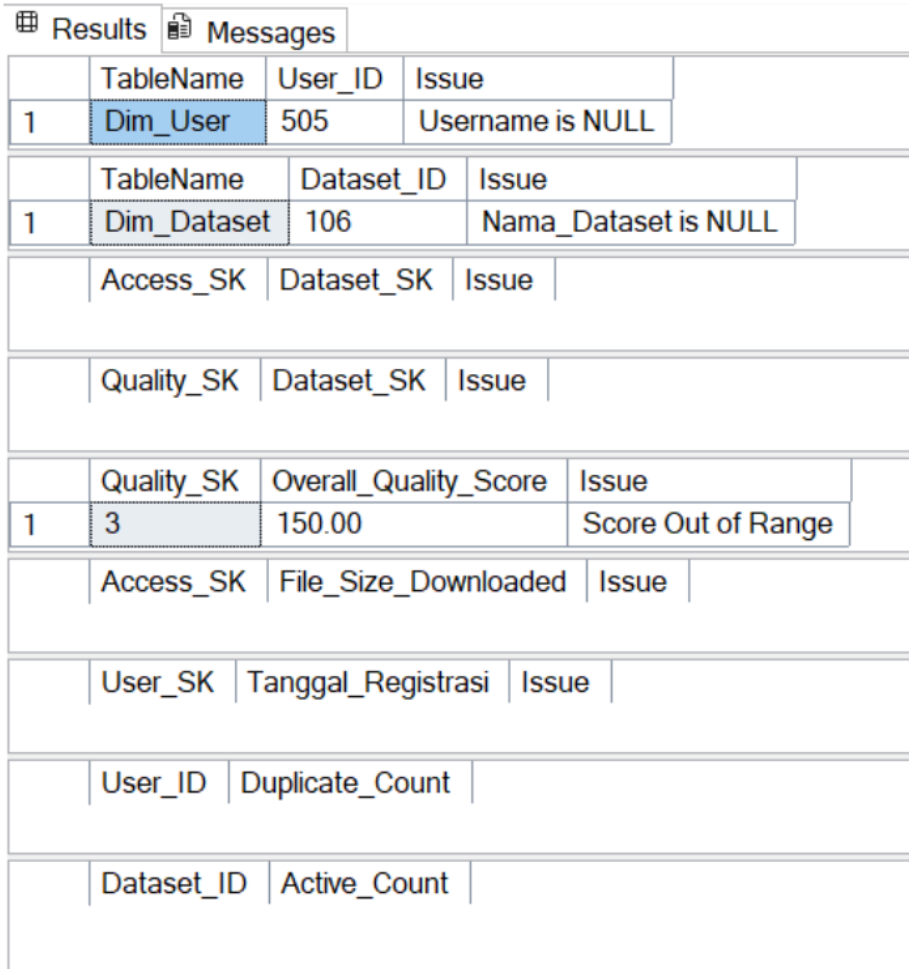
```
LEFT JOIN dbo.Dim_User u ON s.User_ID = u.User_ID
WHERE d.Dataset_SK IS NOT NULL; -- Filter Data Kotor
```

Listing 3: Potongan Logika ETL Fact Access

4 Quality Assurance (QA)

Untuk memastikan data yang masuk berkualitas, kami melakukan pengujian dengan menyuntikkan data "sampah" (invalid) ke dalam sistem.

- **Skenario Uji:** Kami mencoba memasukkan data dengan Skor Kualitas di atas 100, data User tanpa nama (NULL), dan nama dataset yang bernilai NULL.
- **Hasil:** Script *Quality Check* kami berhasil mendeteksi anomali tersebut dan melaporkannya sebagai error.



Gambar 3: Hasil Deteksi Data Anomali oleh Script Quality Check

5 Laporan Performa

Setelah dilakukan pengujian beban (*stress test*) menggunakan 10.000+ data dummy, sistem menunjukkan performa yang sangat memuaskan. Berkat strategi Indexing dan Partitioning, waktu eksekusi query rata-rata sangat singkat.

Tabel 1: Hasil Benchmark Performa Query

Jenis Query	**Target**	**Aktual**	**Status**
Simple Aggregation	$<$ 1 detik	5 ms	**Pass**
Complex Join	$<$ 3 detik	8 ms	**Pass**
Drill-down Analysis	$<$ 2 detik	4 ms	**Pass**
Full Scan	$<$ 10 detik	15 ms	**Pass**