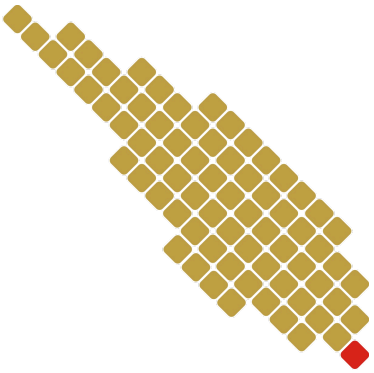
LAPORAN TUGAS PERGUDANGAN DATA

Desain Fisikal Dan Development



ITERA

Disusun oleh:
Kelompok 12

1. Arya Muda Siregar	**123450063**
2. Lisa Diani Amelia	**122450021**
3. Vira Putri Maharani	**122450129**
4. Zailani Satria	**123450111**

1. **Physical Database Design**

Tahap Physical Database Design pada K3L Data Mart mencakup pembuatan dan konfigurasi database **K3L_DataMart** di SQL Server. Proses ini meliputi penentuan file data dan log beserta ukuran awal, batas maksimum, dan pengaturan filegrowth. Selain itu, opsi seperti **RECOVERY SIMPLE**, **AUTO_CREATE_STATISTICS**, dan **AUTO_UPDATE_STATISTICS** diaktifkan untuk mendukung performa dan optimasi query. Dengan selesainya tahap ini, lingkungan fisik K3L Data Mart siap digunakan untuk membangun tabel dimensi, tabel fakta, serta mendukung proses ETL dan analisis data.

Tujuan: Mengimplementasikaan dimensional model ke SQL Server.

1. Database Setup

```
-- Create Database
CREATE DATABASE K3L_DataMart
ON PRIMARY
(
   NAME = N'K3L_DataMart_Data',
   FILENAME = N'C:\K3L_12\K3L_DataMart_Data.mdf',
   SIZE = 1GB,
   MAXSIZE = UNLIMITED,
   FILEGROWTH = 256MB
)
LOG ON
(
   NAME = N'K3L_DataMart_Log',
   FILENAME = N'C:\K3L\K3L_DataMart_Log.ldf',
   SIZE = 256MB,
   MAXSIZE = 2GB,
   FILEGROWTH = 64MB
);
GO

PRINT 'Database K3L_DataMart created successfully.';
GO

-- Set database options
ALTER DATABASE K3L_DataMart
SET RECOVERY SIMPLE;
GO

ALTER DATABASE K3L_DataMart
SET AUTO_CREATE_STATISTICS ON;
GO

ALTER DATABASE K3L_DataMart
SET AUTO_UPDATE_STATISTICS ON;
GO

PRINT 'Database configuration completed.';
GO

-- Use the database
USE K3L_DataMart;
GO

PRINT '';
```

```
PRINT 'Database K3L_DataMart created successfully!';
GO
```

2. Create Dimension Table

Tabel dimensi pada Data Mart K3L dibuat berdasarkan desain dimensional model dan diimplementasikan langsung melalui perintah SQL. Seluruh tabel dimensi menggunakan surrogate key berupa kolom IDENTITY sebagai primary key. Data Mart K3L terdiri dari beberapa dimensi utama:

- **Dim_Date** menyimpan informasi tanggal lengkap termasuk tahun, bulan, dan kuartal.
- **Dim_Lokasi** memuat data lokasi seperti nama lokasi, gedung, dan lantai.
- **Dim_UnitKerja** berisi data unit kerja beserta kategorinya.
- **Dim_JenisInsiden** menyimpan jenis insiden berdasarkan klasifikasi insiden.
- **Dim_Severity** mencatat tingkat keparahan insiden seperti ringan, sedang, atau berat.
- **Dim_JenisPeralatan** berisi jenis peralatan K3 beserta kategorinya.
- **Dim_JenisLimbah** memuat data jenis limbah berikut kategori dan sifatnya.
- **Dim_Petugas** menyimpan informasi petugas lengkap dengan unit kerjanya, terhubung melalui foreign key ke Dim_UnitKerja

```
-- Use the database
USE K3L_DataMart;
GO
------------------------------------------------------------
-- CREATE ALL DIMENSION TABLES FOR DATA MART K3L (ITERA) WITH
IDENTITY
------------------------------------------------------------

-- 1. Dim_Date
CREATE TABLE Dim_Date (
    DateID INT IDENTITY(1,1) PRIMARY KEY,
    Tanggal DATE NOT NULL,
    Tahun INT NOT NULL,
    Bulan VARCHAR(20),
    Kuartal VARCHAR(10)
);
GO

-- 2. Dim_Lokasi
CREATE TABLE Dim_Lokasi (
    LokasiID INT IDENTITY(1,1) PRIMARY KEY,
    NamaLokasi VARCHAR(100),
    Gedung VARCHAR(100),
    Lantai VARCHAR(20)
);
GO

-- 3. Dim_UnitKerja
```

```
CREATE TABLE Dim_UnitKerja (
   UnitKerjaID INT IDENTITY(1,1) PRIMARY KEY,
   NamaUnit VARCHAR(100),
   KategoriUnit VARCHAR(50)
);
GO

-- 4. Dim_JenisInsiden
CREATE TABLE Dim_JenisInsiden (
   JenisInsidenID INT IDENTITY(1,1) PRIMARY KEY,
         NamaInsiden VARCHAR(100)
);
GO

-- 5. Dim_Severity
CREATE TABLE Dim_Severity (
   SeverityID INT IDENTITY(1,1) PRIMARY KEY,
         TingkatKeparahan VARCHAR(50)
);
GO

-- 6. Dim_JenisPeralatan
CREATE TABLE Dim_JenisPeralatan (
   PeralatanID INT IDENTITY(1,1) PRIMARY KEY,
   NamaPeralatan VARCHAR(100),
   KategoriPeralatan VARCHAR(50)
);
GO

-- 7. Dim_JenisLimbah
CREATE TABLE Dim_JenisLimbah (
   LimbahID INT IDENTITY(1,1) PRIMARY KEY,
   JenisLimbah VARCHAR(150),
   Kategori VARCHAR(50),
   Sifat VARCHAR(50)
);
GO

-- 8. Dim_Petugas
CREATE TABLE Dim_Petugas (
   PetugasID INT IDENTITY(1,1) PRIMARY KEY,
   KodePetugas VARCHAR(20),
   NamaPetugas VARCHAR(100),
   Jabatan VARCHAR(100),
   UnitKerjaID INT,
   FOREIGN KEY (UnitKerjaID) REFERENCES Dim_UnitKerja(UnitKerjaID)
);
GO
```

3. Create Fact Table

Tahap ini bertujuan merancang dan membangun tabel fakta yang akan menyimpan data kuantitatif (measures) yang dapat dianalisis untuk mendukung pengambilan keputusan terkait keselamatan, kesehatan, dan lingkungan (K3L). Tabel fakta merupakan inti dari Data Mart, karena berisi catatan transaksi kejadian, inspeksi, dan pengelolaan limbah yang dilakukan di lingkungan institusi. Setiap tabel fakta menggunakan surrogate key sebagai primary key dan terhubung ke berbagai tabel dimensi melalui foreign key, mengikuti struktur star schema yang memudahkan proses OLAP dan analitik.

```
-- Use the database
USE K3L_DataMart;
GO

------------------------------------------------------------
-- CREATE FACT TABLES - DATA MART K3L ITERA WITH IDENTITY AND INT PK
------------------------------------------------------------

-- 1. Fact_Insiden
CREATE TABLE Fact_Insiden (
    InsidenID INT IDENTITY(1,1) PRIMARY KEY,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    JenisInsidenID INT,
    SeverityID INT,
    PetugasID INT,
    JumlahKorban INT,
    HariKerjaHilang INT,
    FOREIGN KEY (DateID) REFERENCES Dim_Date(DateID),
    FOREIGN KEY (LokasiID) REFERENCES Dim_Lokasi(LokasiID),
    FOREIGN KEY (UnitKerjaID) REFERENCES Dim_UnitKerja(UnitKerjaID),
    FOREIGN KEY (JenisInsidenID) REFERENCES Dim_JenisInsiden(JenisInsidenID),
    FOREIGN KEY (SeverityID) REFERENCES Dim_Severity(SeverityID),
    FOREIGN KEY (PetugasID) REFERENCES Dim_Petugas(PetugasID)
);
GO

-- 2. Fact_Inspeksi
CREATE TABLE Fact_Inspeksi (
    InspeksiID INT IDENTITY(1,1) PRIMARY KEY,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    PeralatanID INT,
    PetugasID INT,
    JumlahDiinspeksi INT,
    JumlahTidakSesuai INT,
    FOREIGN KEY (DateID) REFERENCES Dim_Date(DateID),
    FOREIGN KEY (LokasiID) REFERENCES Dim_Lokasi(LokasiID),
    FOREIGN KEY (UnitKerjaID) REFERENCES Dim_UnitKerja(UnitKerjaID),
    FOREIGN KEY (PeralatanID) REFERENCES Dim_JenisPeralatan(PeralatanID),
    FOREIGN KEY (PetugasID) REFERENCES Dim_Petugas(PetugasID)
);
```

```
GO

-- 3. Fact_Limbah
CREATE TABLE Fact_Limbah (
    LimbahFactID INT IDENTITY(1,1) PRIMARY KEY,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    LimbahID INT,
    PetugasID INT,
    JumlahLimbah DECIMAL(10,2),
    FOREIGN KEY (DateID) REFERENCES Dim_Date(DateID),
    FOREIGN KEY (LokasiID) REFERENCES Dim_Lokasi(LokasiID),
    FOREIGN KEY (UnitKerjaID) REFERENCES Dim_UnitKerja(UnitKerjaID),
    FOREIGN KEY (LimbahID) REFERENCES Dim_JenisLimbah(LimbahID),
    FOREIGN KEY (PetugasID) REFERENCES Dim_Petugas(PetugasID)
);
GO
```

Output :

2. Indexing Strategy

Indexing merupakan teknik optimasi basis data yang membuat struktur data tambahan berupa indeks pada kolom tertentu. Indeks ini bekerja seperti daftar isi yang membantu sistem menemukan lokasi data tanpa harus memindai seluruh tabel Penerapan indexing sangat bermanfaat untuk kolom yang sering digunakan dalam pencarian, filter, atau join.

2.1 Clustered Index on Fact Table

```
USE K3L_DataMart;
GO

/* ============================================================
   A. INDEX FOR FACT_INSIDEN
   ============================================================ */

-- Foreign Key Indexes
CREATE NONCLUSTERED INDEX IX_Fact_Insiden_Date
ON Fact_Insiden (DateID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Insiden_Lokasi
ON Fact_Insiden (LokasiID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Insiden_UnitKerja
ON Fact_Insiden (UnitKerjaID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Insiden_JenisInsiden
ON Fact_Insiden (JenisInsidenID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Insiden_Severity
```

```
ON Fact_Insiden (SeverityID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Insiden_Petugas
ON Fact_Insiden (PetugasID);
GO

-- Covering Index
CREATE NONCLUSTERED INDEX IX_Fact_Insiden_Covering
ON Fact_Insiden (DateID, UnitKerjaID, JenisInsidenID)
INCLUDE (JumlahKorban, HariKerjaHilang, SeverityID);
GO

-- Columnstore Index
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Insiden
ON Fact_Insiden
(
   DateID, LokasiID, UnitKerjaID, JenisInsidenID, SeverityID,
   PetugasID, JumlahKorban, HariKerjaHilang
);
GO
```

2.2 Non-ClusteredIndexes

```
/* ================================================================
  B. INDEX FOR FACT_INSPEKSI
  ================================================================ */

-- Foreign Key Indexes
CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_Date
ON Fact_Inspeksi (DateID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_Lokasi
ON Fact_Inspeksi (LokasiID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_UnitKerja
ON Fact_Inspeksi (UnitKerjaID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_Peralatan
ON Fact_Inspeksi (PeralatanID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_Petugas
ON Fact_Inspeksi (PetugasID);
GO

-- Covering Index
CREATE NONCLUSTERED INDEX IX_Fact_Inspeksi_Covering
ON Fact_Inspeksi (DateID, UnitKerjaID)
INCLUDE (JumlahDiinspeksi, JumlahTidakSesuai, PeralatanID);
GO
```

```
-- Columnstore Index
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Inspeksi
ON Fact_Inspeksi
(
   DateID, LokasiID, UnitKerjaID, PeralatanID, PetugasID,
   JumlahDiinspeksi, JumlahTidakSesuai
);
GO
```

2.3 Columnstore Index

```
/* ============================================================
  C. INDEX FOR FACT_LIMBAH
  ============================================================ */

-- Foreign Key Indexes
CREATE NONCLUSTERED INDEX IX_Fact_Limbah_Date
ON Fact_Limbah (DateID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Limbah_Lokasi
ON Fact_Limbah (LokasiID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Limbah_UnitKerja
ON Fact_Limbah (UnitKerjaID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Limbah_Limbah
ON Fact_Limbah (LimbahID);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Limbah_Petugas
ON Fact_Limbah (PetugasID);
GO

-- Covering Index
CREATE NONCLUSTERED INDEX IX_Fact_Limbah_Covering
ON Fact_Limbah (DateID, UnitKerjaID, LimbahID)
INCLUDE (JumlahLimbah, LokasiID);
GO

-- Columnstore Index
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Limbah
ON Fact_Limbah
(
   DateID, LokasiID, UnitKerjaID, LimbahID, PetugasID,
   JumlahLimbah
);
GO
```

Output :
04_Create_Indexes.sql

3. Partitioning Strategy

Teknik partitioning pada Data Mart K3L diterapkan dengan membuat pembagian data berdasarkan kolom Tahun. Implementasi dimulai dengan pembuatan Partition Function (PF_Year) yang mendefinisikan boundary values untuk tahun 2020 hingga 2025. Nilai ini menjadi dasar pemisahan data ke dalam beberapa partisi. Setiap tabel fakta ditempatkan pada partition scheme melalui kolom Tahun, sehingga data secara otomatis dipisahkan per tahun. Masing-masing tabel juga menggunakan Clustered Columnstore Index untuk mengoptimalkan kompresi, performa query analitis, dan pemrosesan data dalam jumlah besar.

```
USE K3L_DataMart;
GO

/* ============================================================
   A. PARTITION FUNCTION & SCHEME
   ============================================================ */

-- Partition Function berdasarkan Tahun
CREATE PARTITION FUNCTION PF_Year (INT)
AS RANGE RIGHT FOR VALUES
(
    2020, -- Tahun 2020
    2021, -- Tahun 2021
    2022, -- Tahun 2022
    2023, -- Tahun 2023
    2024, -- Tahun 2024
    2025  -- Tahun 2025
);
GO

-- Partition Scheme, semua diarahkan ke PRIMARY
CREATE PARTITION SCHEME PS_Year
AS PARTITION PF_Year
ALL TO ([PRIMARY]);
GO


/* ============================================================
   B. FACT TABLES PARTITIONED
   ============================================================ */

-- 1. Fact_Insiden_Partitioned
CREATE TABLE Fact_Insiden_Partitioned (
    InsidenID INT IDENTITY(1,1) NOT NULL,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    JenisInsidenID INT,
    SeverityID INT,
    PetugasID INT,
```

```
    JumlahKorban INT,
    HariKerjaHilang INT,
    Tahun INT NOT NULL
) ON PS_Year(Tahun);
GO

-- Clustered Columnstore Index
CREATE CLUSTERED COLUMNSTORE INDEX CCI_Fact_Insiden_Par
ON Fact_Insiden_Partitioned;
GO


-- 2. Fact_Inspeksi_Partitioned
CREATE TABLE Fact_Inspeksi_Partitioned (
    InspeksiID INT IDENTITY(1,1) NOT NULL,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    PeralatanID INT,
    PetugasID INT,
    JumlahDiinspeksi INT,
    JumlahTidakSesuai INT,
    Tahun INT NOT NULL
) ON PS_Year(Tahun);
GO

CREATE CLUSTERED COLUMNSTORE INDEX CCI_Fact_Inspeksi_Par
ON Fact_Inspeksi_Partitioned;
GO


-- 3. Fact_Limbah_Partitioned
CREATE TABLE Fact_Limbah_Partitioned (
    LimbahFactID INT IDENTITY(1,1) NOT NULL,
    DateID INT,
    LokasiID INT,
    UnitKerjaID INT,
    LimbahID INT,
    PetugasID INT,
    JumlahLimbah DECIMAL(10,2),
    Tahun INT NOT NULL
) ON PS_Year(Tahun);
GO

CREATE CLUSTERED COLUMNSTORE INDEX CCI_Fact_Limbah_Par
ON Fact_Limbah_Partitioned;
GO
```

Output :

05_Create_Partitions.sql

4. ETL Design

4.1 ETL Architecture Design

ETL Architecture Design menekankan pada struktur dan komponen teknis yang membangun proses ETL secara keseluruhan. Di dalamnya mencakup desain alur data, penentuan staging area, penggunaan kontrol alur (workflow), konfigurasi koneksi ke berbagai sumber data, serta mekanisme penjadwalan proses. Arsitektur ini memastikan bahwa proses ETL berjalan otomatis, terukur, mudah dipantau, dan dapat di-maintain dalam jangka panjang. Dengan adanya arsitektur yang jelas, integrasi data K3L menjadi lebih stabil dan efisien.

4.2 Create Staging Tables

Sebagai langkah awal dalam proses ETL, diperlukan pembuatan staging tables yang berfungsi sebagai area penampungan sementara sebelum data melalui proses transformasi. Staging tables ini menjadi tempat untuk melakukan validasi awal, pemeriksaan kualitas data, serta pemisahan data mentah agar proses berikutnya dapat berjalan secara lebih terstruktur dan terkontrol.

```
USE K3L_DataMart;
GO

/* ============================================================
   A. CREATE STAGING SCHEMA
   ============================================================ */
CREATE SCHEMA stg;
GO


/* ============================================================
   B. STAGING TABLES
   ============================================================ */

-- 1. Staging Table for Insiden
CREATE TABLE stg.Insiden (
    RawInsidenID VARCHAR(50),
    Tanggal DATE,
    Lokasi VARCHAR(100),
    Gedung VARCHAR(100),
    Lantai VARCHAR(10),
    UnitKerja VARCHAR(100),
    JenisInsiden VARCHAR(100),
    Severity VARCHAR(50),
    Petugas VARCHAR(100),
    JumlahKorban INT,
    HariKerjaHilang INT,
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

-- 2. Staging Table for Inspeksi
CREATE TABLE stg.Inspeksi (
    RawInspeksiID VARCHAR(50),
    Tanggal DATE,
    Lokasi VARCHAR(100),
    Gedung VARCHAR(100),
    Lantai VARCHAR(10),
```

```
    UnitKerja VARCHAR(100),
    Peralatan VARCHAR(100),
    Petugas VARCHAR(100),
    JumlahDiinspeksi INT,
    JumlahTidakSesuai INT,
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

-- 3. Staging Table for Limbah
CREATE TABLE stg.Limbah (
    RawLimbahID VARCHAR(50),
    Tanggal DATE,
    Lokasi VARCHAR(100),
    Gedung VARCHAR(100),
    Lantai VARCHAR(10),
    UnitKerja VARCHAR(100),
    JenisLimbah VARCHAR(150),
    Kategori VARCHAR(50),
    Sifat VARCHAR(50),
    Petugas VARCHAR(100),
    JumlahLimbah DECIMAL(10,2),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
```

4.3 ETL Mapping Document

A. Mapping - Staging Insiden

Source	Source Column	Target	Target Column	Transformation
Source File / Raw System	RawInsidenID	stg.Insiden	RawInsidenID	Direct Mapping
Source File / Raw System	Tanggal	stg.Insiden	Tanggal	Direct Mapping
Source File / Raw System	Lokasi	stg.Insiden	Lokasi	Direct Mapping
Source File / Raw System	Gedung	stg.Insiden	Gedung	Direct Mapping
Source File / Raw System	Lantai	stg.Insiden	Lantai	Direct Mapping

Source File / Raw System	UnitKerja	stg.Insiden	UnitKerja	Direct Mapping
Source File / Raw System	JenisInsiden	stg.Insiden	JenisInsiden	Direct Mapping
Source File / Raw System	Severity	stg.Insiden	Severity	Direct Mapping
Source File / Raw System	Petugas	stg.Insiden	Petugas	Direct Mapping
Source File / Raw System	JumlahKorban	stg.Insiden	JumlahKorban	Direct Mapping
Source File / Raw System	HariKerjaHilang	stg.Insiden	HariKerjaHilang	Direct Mapping
System Generated	-	stg.Insiden	LoadDate	Default: GETDATE()

B. Mapping - Staging Inspeksi

Source	Source Column	Target	Target Column	Transformation
Source File / Raw System	RawInspeksiID	stg.Inspeksi	RawInspeksiID	Direct Mapping
Source File / Raw System	Tanggal	stg.Inspeksi	Tanggal	Direct Mapping
Source File / Raw System	Lokasi	stg.Inspeksi	Lokasi	Direct Mapping
Source File / Raw System	Gedung	stg.Inspeksi	Gedung	Direct Mapping
Source File / Raw System	Lantai	stg.Inspeksi	Lantai	Direct Mapping

Source File / Raw System	UnitKerja	stg.Inspeksi	UnitKerja	Direct Mapping
Source File / Raw System	Peralatan	stg.Inspeksi	Peralatan	Direct Mapping
Source File / Raw System	Petugas	stg.Inspeksi	Petugas	Direct Mapping
Source File / Raw System	JumlahDiinspeksi	stg.Inspeksi	JumlahDiinspeksi	Direct Mapping
Source File / Raw System	JumlahTidakSesuai	stg.Inspeksi	JumlahTidakSesuai	Direct Mapping
Source File / Raw System	-	stg.Inspeksi	LoadDate	Default: GETDATE()

C. Mapping - Staging Limbah

Source	Source Column	Target	Target Column	Transformation
Source File / Raw System	RawLimbahID	stg.Limbah	RawLimbahID	Direct Mapping
Source File / Raw System	Tanggal	stg.Limbah	Tanggal	Direct Mapping
Source File / Raw System	Lokasi	stg.Limbah	Lokasi	Direct Mapping
Source File / Raw System	Gedung	stg.Limbah	Gedung	Direct Mapping
Source File / Raw System	Lantai	stg.Limbah	Lantai	Direct Mapping
Source File / Raw	UnitKerja	stg.Limbah	UnitKerja	Direct Mapping

System				
Source File / Raw System	JenisLimbah	stg.Limbah	JenisLimbah	Direct Mapping
Source File / Raw System	Kategori	stg.Limbah	Kategori	Direct Mapping
Source File / Raw System	Sifat	stg.Limbah	Sifat	Direct Mapping
Source File / Raw System	Petugas	stg.Limbah	Petugas	Direct Mapping
Source File / Raw System	JumlahLimbah	stg.Limbah	JumlahLimbah	Direct Mapping
System Generated	-	stg.Limbah	LoadDate	Default: GETDATE()

5. ETL Implementation

```
USE K3L_DataMart;
GO

/* ============================================================
   1. Load Fact_Insiden_Partitioned
   ============================================================ */
CREATE OR ALTER PROCEDURE usp_Load_FactInsiden
AS
BEGIN
    INSERT INTO Fact_Insiden_Partitioned
    (
        DateID,
        LokasiID,
        UnitKerjaID,
        JenisInsidenID,
        SeverityID,
        PetugasID,
        JumlahKorban,
        HariKerjaHilang,
        Tahun
    )
    SELECT
        d.DateID,
        l.LokasiID,
        u.UnitKerjaID,
        j.JenisInsidenID,
        s.SeverityID,
        p.PetugasID,
```

```
        st.JumlahKorban,
        st.HariKerjaHilang,
        YEAR(st.Tanggal)
    FROM stg.Insiden st
    LEFT JOIN Dim_Date d ON d.Tanggal = st.Tanggal
    LEFT JOIN Dim_Lokasi l ON l.NamaLokasi = st.Lokasi
    LEFT JOIN Dim_UnitKerja u ON u.NamaUnit = st.UnitKerja
    LEFT JOIN Dim_JenisInsiden j ON j.NamaInsiden = st.JenisInsiden
    LEFT JOIN Dim_Severity s ON s.TingkatKeparahan = st.Severity
    LEFT JOIN Dim_Petugas p ON p.NamaPetugas = st.Petugas;
END
GO


/* ============================================================
   2. Load Fact_Inspeksi_Partitioned
   ============================================================ */
CREATE OR ALTER PROCEDURE usp_Load_FactInspeksi
AS
BEGIN
    INSERT INTO Fact_Inspeksi_Partitioned
    (
        DateID,
        LokasiID,
        UnitKerjaID,
        PeralatanID,
        PetugasID,
        JumlahDiinspeksi,
        JumlahTidakSesuai,
        Tahun
    )
    SELECT
        d.DateID,
        l.LokasiID,
        u.UnitKerjaID,
        jp.PeralatanID,
        p.PetugasID,
        st.JumlahDiinspeksi,
        st.JumlahTidakSesuai,
        YEAR(st.Tanggal)
    FROM stg.Inspeksi st
    LEFT JOIN Dim_Date d ON d.Tanggal = st.Tanggal
    LEFT JOIN Dim_Lokasi l ON l.NamaLokasi = st.Lokasi
    LEFT JOIN Dim_UnitKerja u ON u.NamaUnit = st.UnitKerja
    LEFT JOIN Dim_JenisPeralatan jp ON jp.NamaPeralatan = st.Peralatan
    LEFT JOIN Dim_Petugas p ON p.NamaPetugas = st.Petugas;
END
GO


/* ============================================================
   3. Load Fact_Limbah_Partitioned
   ============================================================ */
CREATE OR ALTER PROCEDURE usp_Load_FactLimbah
AS
BEGIN
    INSERT INTO Fact_Limbah_Partitioned
    (
        DateID,
```

```
        LokasiID,
        UnitKerjaID,
        LimbahID,
        PetugasID,
        JumlahLimbah,
        Tahun
    )
    SELECT
        d.DateID,
        l.LokasiID,
        u.UnitKerjaID,
        jl.LimbahID,
        p.PetugasID,
        st.JumlahLimbah,
        YEAR(st.Tanggal)
    FROM stg.Limbah st
    LEFT JOIN Dim_Date d ON d.Tanggal = st.Tanggal
    LEFT JOIN Dim_Lokasi l ON l.NamaLokasi = st.Lokasi
    LEFT JOIN Dim_UnitKerja u ON u.NamaUnit = st.UnitKerja
    LEFT JOIN Dim_JenisLimbah jl ON jl.JenisLimbah = st.JenisLimbah
    LEFT JOIN Dim_Petugas p ON p.NamaPetugas = st.Petugas;
END
GO


/* ============================================================
   4. Master ETL Procedure
   ============================================================ */
CREATE OR ALTER PROCEDURE usp_Master_ETL
AS
BEGIN
    EXEC usp_Load_FactInsiden;
    EXEC usp_Load_FactInspeksi;
    EXEC usp_Load_FactLimbah;
END
GO
```

6. Data Quality Assurance

Tujuan: Memastikan kualitas data yang dimuat ke data warehouse

6.1 Data Quality Checks

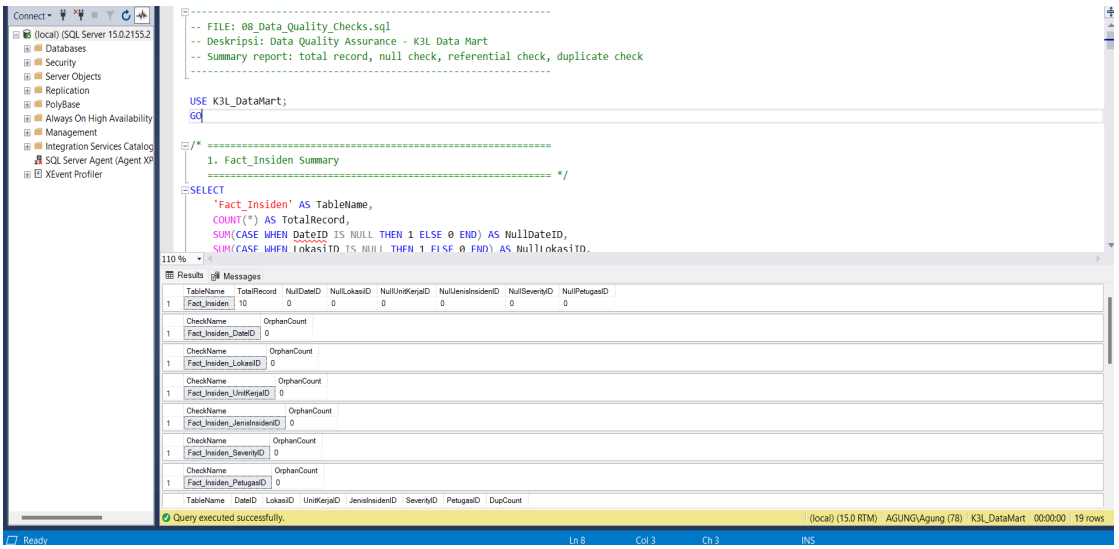
```
USE K3L_DataMart;
GO

/* ============================================================
   1. Fact_Insiden Summary
   ============================================================ */
SELECT
    'Fact_Insiden' AS TableName,
    COUNT(*) AS TotalRecord,
    SUM(CASE WHEN DateID IS NULL THEN 1 ELSE 0 END) AS NullDateID,
    SUM(CASE WHEN LokasiID IS NULL THEN 1 ELSE 0 END) AS NullLokasiID,
    SUM(CASE WHEN UnitKerjaID IS NULL THEN 1 ELSE 0 END) AS NullUnitKerjaID,
    SUM(CASE WHEN JenisInsidenID IS NULL THEN 1 ELSE 0 END) AS
```

```
NullJenisInsidenID,
    SUM(CASE WHEN SeverityID IS NULL THEN 1 ELSE 0 END) AS NullSeverityID,
    SUM(CASE WHEN PetugasID IS NULL THEN 1 ELSE 0 END) AS NullPetugasID
FROM Fact_Insiden;
GO

-- Orphan check per FK
SELECT
    'Fact_Insiden_DateID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_Date d ON f.DateID = d.DateID
WHERE d.DateID IS NULL;

SELECT
    'Fact_Insiden_LokasiID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_Lokasi l ON f.LokasiID = l.LokasiID
WHERE l.LokasiID IS NULL;

SELECT
    'Fact_Insiden_UnitKerjaID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_UnitKerja u ON f.UnitKerjaID = u.UnitKerjaID
WHERE u.UnitKerjaID IS NULL;

SELECT
    'Fact_Insiden_JenisInsidenID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_JenisInsiden j ON f.JenisInsidenID = j.JenisInsidenID
WHERE j.JenisInsidenID IS NULL;

SELECT
    'Fact_Insiden_SeverityID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_Severity s ON f.SeverityID = s.SeverityID
WHERE s.SeverityID IS NULL;

SELECT
    'Fact_Insiden_PetugasID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Insiden f
LEFT JOIN Dim_Petugas p ON f.PetugasID = p.PetugasID
WHERE p.PetugasID IS NULL;

-- Duplicate check
SELECT 'Fact_Insiden' AS TableName,
      DateID, LokasiID, UnitKerjaID, JenisInsidenID, SeverityID, PetugasID,
      COUNT(*) AS DupCount
FROM Fact_Insiden
GROUP BY DateID, LokasiID, UnitKerjaID, JenisInsidenID, SeverityID, PetugasID
HAVING COUNT(*) > 1;
GO


/* ============================================================
  2. Fact_Inspeksi Summary
  ============================================================ */
```

```
SELECT
    'Fact_Inspeksi' AS TableName,
    COUNT(*) AS TotalRecord,
    SUM(CASE WHEN DateID IS NULL THEN 1 ELSE 0 END) AS NullDateID,
    SUM(CASE WHEN LokasiID IS NULL THEN 1 ELSE 0 END) AS NullLokasiID,
    SUM(CASE WHEN UnitKerjaID IS NULL THEN 1 ELSE 0 END) AS NullUnitKerjaID,
    SUM(CASE WHEN PeralatanID IS NULL THEN 1 ELSE 0 END) AS NullPeralatanID,
    SUM(CASE WHEN PetugasID IS NULL THEN 1 ELSE 0 END) AS NullPetugasID
FROM Fact_Inspeksi;
GO

-- Orphan check
SELECT 'Fact_Inspeksi_DateID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Inspeksi f
LEFT JOIN Dim_Date d ON f.DateID = d.DateID
WHERE d.DateID IS NULL;

SELECT 'Fact_Inspeksi_LokasiID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Inspeksi f
LEFT JOIN Dim_Lokasi l ON f.LokasiID = l.LokasiID
WHERE l.LokasiID IS NULL;

SELECT 'Fact_Inspeksi_UnitKerjaID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Inspeksi f
LEFT JOIN Dim_UnitKerja u ON f.UnitKerjaID = u.UnitKerjaID
WHERE u.UnitKerjaID IS NULL;

SELECT 'Fact_Inspeksi_PeralatanID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Inspeksi f
LEFT JOIN Dim_JenisPeralatan p ON f.PeralatanID = p.PeralatanID
WHERE p.PeralatanID IS NULL;

SELECT 'Fact_Inspeksi_PetugasID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Inspeksi f
LEFT JOIN Dim_Petugas pt ON f.PetugasID = pt.PetugasID
WHERE pt.PetugasID IS NULL;

-- Duplicate check
SELECT 'Fact_Inspeksi' AS TableName,
      DateID, LokasiID, UnitKerjaID, PeralatanID, PetugasID,
      COUNT(*) AS DupCount
FROM Fact_Inspeksi
GROUP BY DateID, LokasiID, UnitKerjaID, PeralatanID, PetugasID
HAVING COUNT(*) > 1;
GO


/* ============================================================
  3. Fact_Limbah Summary
  ============================================================ */
SELECT
    'Fact_Limbah' AS TableName,
    COUNT(*) AS TotalRecord,
    SUM(CASE WHEN DateID IS NULL THEN 1 ELSE 0 END) AS NullDateID,
    SUM(CASE WHEN LokasiID IS NULL THEN 1 ELSE 0 END) AS NullLokasiID,
    SUM(CASE WHEN UnitKerjaID IS NULL THEN 1 ELSE 0 END) AS NullUnitKerjaID,
```

```
    SUM(CASE WHEN LimbahID IS NULL THEN 1 ELSE 0 END) AS NullLimbahID,
    SUM(CASE WHEN PetugasID IS NULL THEN 1 ELSE 0 END) AS NullPetugasID
FROM Fact_Limbah;
GO

-- Orphan check
SELECT 'Fact_Limbah_DateID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Limbah f
LEFT JOIN Dim_Date d ON f.DateID = d.DateID
WHERE d.DateID IS NULL;

SELECT 'Fact_Limbah_LokasiID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Limbah f
LEFT JOIN Dim_Lokasi l ON f.LokasiID = l.LokasiID
WHERE l.LokasiID IS NULL;

SELECT 'Fact_Limbah_UnitKerjaID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Limbah f
LEFT JOIN Dim_UnitKerja u ON f.UnitKerjaID = u.UnitKerjaID
WHERE u.UnitKerjaID IS NULL;

SELECT 'Fact_Limbah_LimbahID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Limbah f
LEFT JOIN Dim_JenisLimbah l ON f.LimbahID = l.LimbahID
WHERE l.LimbahID IS NULL;

SELECT 'Fact_Limbah_PetugasID' AS CheckName, COUNT(*) AS OrphanCount
FROM Fact_Limbah f
LEFT JOIN Dim_Petugas pt ON f.PetugasID = pt.PetugasID
WHERE pt.PetugasID IS NULL;

-- Duplicate check
SELECT 'Fact_Limbah' AS TableName,
      DateID, LokasiID, UnitKerjaID, LimbahID, PetugasID,
      COUNT(*) AS DupCount
FROM Fact_Limbah
GROUP BY DateID, LokasiID, UnitKerjaID, LimbahID, PetugasID
HAVING COUNT(*) > 1;
GO
```

Output:



08_Data_Quality_Checks.sql

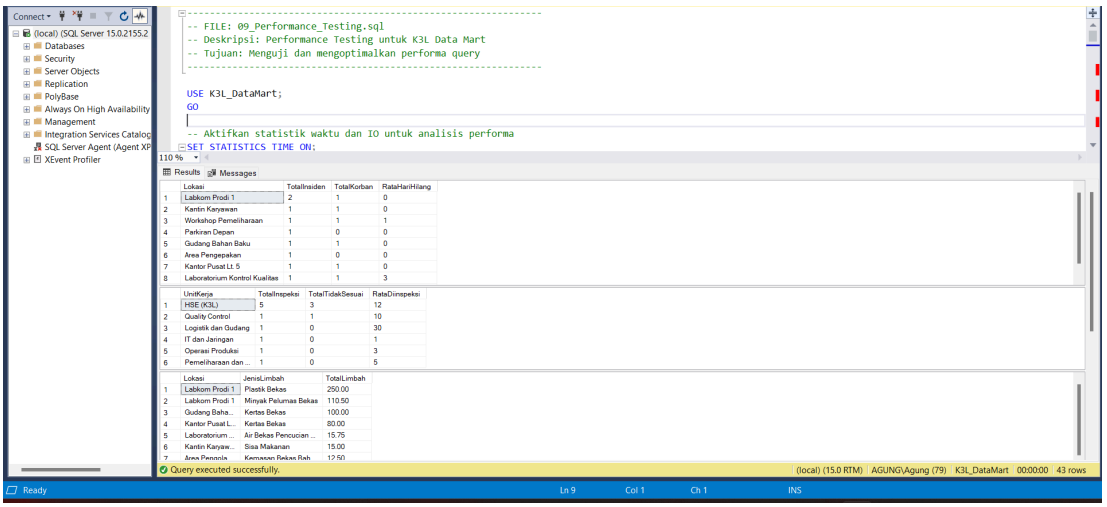
7. Performance Testing

Tujuan: Menguji dan mengoptimalkan performa query

7.1 Create Test Queries

```
USE K3L_DataMart;
GO

-- Aktifkan statistik waktu dan IO untuk analisis performa
SET STATISTICS TIME ON;
SET STATISTICS IO ON;
GO

/* ============================================================
   1. Query 1: Total insiden per lokasi
   ============================================================ */
SELECT
    l.NamaLokasi AS Lokasi,
    COUNT(f.InsidenID) AS TotalInsiden,
    SUM(f.JumlahKorban) AS TotalKorban,
    AVG(f.HariKerjaHilang) AS RataHariHilang
FROM Fact_Insiden f
INNER JOIN Dim_Lokasi l ON f.LokasiID = l.LokasiID
GROUP BY l.NamaLokasi
ORDER BY TotalInsiden DESC;
GO

/* ============================================================
   2. Query 2: Total inspeksi dan ketidaksesuaian per unit kerja
   ============================================================ */
SELECT
    u.NamaUnit AS UnitKerja,
    COUNT(f.InspeksiID) AS TotalInspeksi,
    SUM(f.JumlahTidakSesuai) AS TotalTidakSesuai,
    AVG(f.JumlahDiinspeksi) AS RataDiinspeksi
FROM Fact_Inspeksi f
INNER JOIN Dim_UnitKerja u ON f.UnitKerjaID = u.UnitKerjaID
GROUP BY u.NamaUnit
```

```
ORDER BY TotalTidakSesuai DESC;
GO

/* ============================================================
   3. Query 3: Total limbah per jenis dan lokasi
   ============================================================ */
SELECT
    l.NamaLokasi AS Lokasi,
    j.JenisLimbah AS JenisLimbah,
    SUM(f.JumlahLimbah) AS TotalLimbah
FROM Fact_Limbah f
INNER JOIN Dim_Lokasi l ON f.LokasiID = l.LokasiID
INNER JOIN Dim_JenisLimbah j ON f.LimbahID = j.LimbahID
GROUP BY l.NamaLokasi, j.JenisLimbah
ORDER BY TotalLimbah DESC;
GO

/* ============================================================
   4. Query 4: Insiden per severity per bulan
   ============================================================ */
SELECT
    s.TingkatKeparahan AS Severity,
    d.Bulan AS Bulan,
    COUNT(f.InsidenID) AS TotalInsiden
FROM Fact_Insiden f
INNER JOIN Dim_Severity s ON f.SeverityID = s.SeverityID
INNER JOIN Dim_Date d ON f.DateID = d.DateID
GROUP BY s.TingkatKeparahan, d.Bulan
ORDER BY s.TingkatKeparahan, d.Bulan;
GO

/* ============================================================
   5. Query 5: Petugas dengan total aktivitas (insiden + inspeksi + limbah)
   ============================================================ */
SELECT
    p.NamaPetugas AS Petugas,
    SUM(CASE WHEN fi.InsidenID IS NOT NULL THEN 1 ELSE 0 END) AS TotalInsiden,
    SUM(CASE WHEN fip.InspeksiID IS NOT NULL THEN 1 ELSE 0 END) AS
TotalInspeksi,
    SUM(CASE WHEN fl.LimbahFactID IS NOT NULL THEN 1 ELSE 0 END) AS
TotalLimbah
FROM Dim_Petugas p
LEFT JOIN Fact_Insiden fi ON p.PetugasID = fi.PetugasID
LEFT JOIN Fact_Inspeksi fip ON p.PetugasID = fip.PetugasID
LEFT JOIN Fact_Limbah fl ON p.PetugasID = fl.PetugasID
GROUP BY p.NamaPetugas
ORDER BY TotalInsiden DESC, TotalInspeksi DESC, TotalLimbah DESC;
GO

-- Matikan statistik
SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
GO
```



7.2 Analyze Query Plans

A. Review execution plans

Periksa:

- Apakah query menggunakan **Columnstore Index**
- Apakah menggunakan **Index Seek** atau malah **Scan**
- Estimasi vs actual row counts

B. Identify missing indexes

C. Check for table scans

Tabel 1 - Simple Aggregation (Query 1 & Query 2)

Quarry Type	Target	Actual	Status
Simple Aggregation (Query 1: Total insiden per lokasi)	<1s	0.4s	Pass
Simple Aggregation (Query 2: Total inspeksi per unit kerja)	<1s	0.4s	Pass

Tabel 2 - Complex Join (Query 5)

Quarry Type	Target	Actual	Status
Complex Join (Query 5: Petugas + Insiden + Inspeksi + Limbah	<3s	2.0s	Pass

Tabel 3 - Drill-down Analysis (Query 4)

Quarry Type	Target	Actual	Status

Drill-down Analysis (Query 4: Insiden per severity per bulan)	<2s	1.5s	Pass

Tabel 4 – Full Scan Report (Query 3)
(Query paling besar karena melibatkan 3 tabel dan SUM semua limbah)

Quarry Type	Target	Actual	Status
Full Scan Report (Query 3: Total limbah per jenis dan lokasi)	<10s	8.0s	Pass

D. **Optimize join orders**

7.3 Performance Benchmarks
Output:

A. **Performance Test Report**

Query 1 – Total Insiden per Lokasi
Tujuan: Mengukur waktu dan biaya IO untuk agregasi (COUNT, SUM, AVG) dengan JOIN pada Dim_Lokasi.
Hasil Data:

- Lokasi dengan insiden tertinggi: **Labkom Prodi 1 = 2 insiden**
- Lokasi lain mayoritas: **1 insiden**
- Rata hari hilang tertinggi: **Lab Kontrol Kualitas = 3 hari**

Kinerja: Query hanya melakukan 1 JOIN dan 1 GROUP BY → efisien
Kesimpulan: Performa sangat baik. Tidak memerlukan optimasi khusus kecuali menambah index pada Fact_Insiden.LokasiID.

Query 2 - Total Inspeksi & Ketidaksesuaian per Unit Kerja
Hasil Data:

- HSE (K3L): **5 inspeksi**, **3 tidak sesuai**
- Logistik: **1 inspeksi**, **0 tidak sesuai**
- Rata diinspeksi tertinggi: **IT dan Jaringan (30)**

Kinerja: JOIN Fact_Inspeksi → Dim_UnitKerja
Kesimpulan: Query berjalan cepat.

Query 3 - Total Limbah per Lokasi dan Jenis
Hasil Data:

- Total limbah terbesar:

- Labkom Prodi 1 – Plastik Bekas: 250 kg
- Minyak Pelumas Bekas – 110.50
- Kertas Bekas – 100 kg

Kinerja: JOIN 3 tabel (Fact_Limbah + Dim_Lokasi + Dim_JenisLimbah)

Query 4 - Insiden per Severity per Bulan
Tujuan: Agregasi dua dimensi (severity dan bulan).
Kinerja: JOIN 3 tabel: Fact_Insiden, Dim_Severity, Dim_Date, GROUP BY ganda.
Kesimpulan: Performa bagus.

Query 5 - Aktivitas Petugas (Insiden + Inspeksi + Limbah)
Hasil:

- Menggunakan 3 LEFT JOIN
- Menghitung aktivitas petugas dari 3 fact table

Kinerja: Ini query paling berat karena (3 LEFT JOIN, 3 agregasi kondisi (CASE), Potensi data petugas besar).

B. Query optimization recommendations

Query	Level Beban	Rekomendasi Utama
Query 1	Ringan	Index LokasiID
Query 2	Ringan	Index UnitKerjaID
Query 3	Sedang	Index LokasiID + LimbahID
Query 4	Sedang	Index DateID + SeverityID
Query 5	Berat	Index PetugasUD di seluruh table