Final Report Sistem Data Warehouse Sarpras
(DM_SARPRAS_DW)

Disusun oleh:

1. Efi Defiyati (123450005)
2. Muhammad Aqil Ramadhan (123450066)
3. Tobias David Manogari (122450091)
4. Cindy Laura Manik (123450112)

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

1. Pendahuluan

1.1. Latar Belakang

Sistem sarana dan prasarana (SARPRAS) memerlukan pengelolaan data inventaris, ruang, dan permintaan fasilitas secara efisien. Sistem DW memungkinkan integrasi data historis dari berbagai sumber sehingga memudahkan analisis, pelaporan, dan pengambilan keputusan.

1.2. Rumusan Masalah

- Bagaimana merancang DW SARPRAS yang terstruktur?
- Bagaimana menyusun dimensi dan fakta terbaru agar analisis optimal?
- Bagaimana mengimplementasikan dan menguji kualitas data DW?

1.3. Tujuan Penelitian

- Merancang DW SARPRAS dengan dimensi dan fakta lengkap.
- Membuat tabel sesuai Database Setup terbaru.
- Menyediakan query analisis yang akurat dan cepat.

1.4. Manfaat Penelitian

- Mempermudah analisis data inventaris, ruang, dan perbaikan.
- Mendukung pengambilan keputusan berbasis data historis.
- Memberikan referensi implementasi DW untuk sistem serupa.

2. Metodologi Perancangan DW

2.1. Analisis Kebutuhan Data

Data berasal dari inventaris, ruang, unit, permintaan fasilitas, dan perbaikan. Analisis dilakukan untuk menentukan fakta dan dimensi relevan.

2.2. Perancangan Dimensi dan Fakta

Dimensi utama:

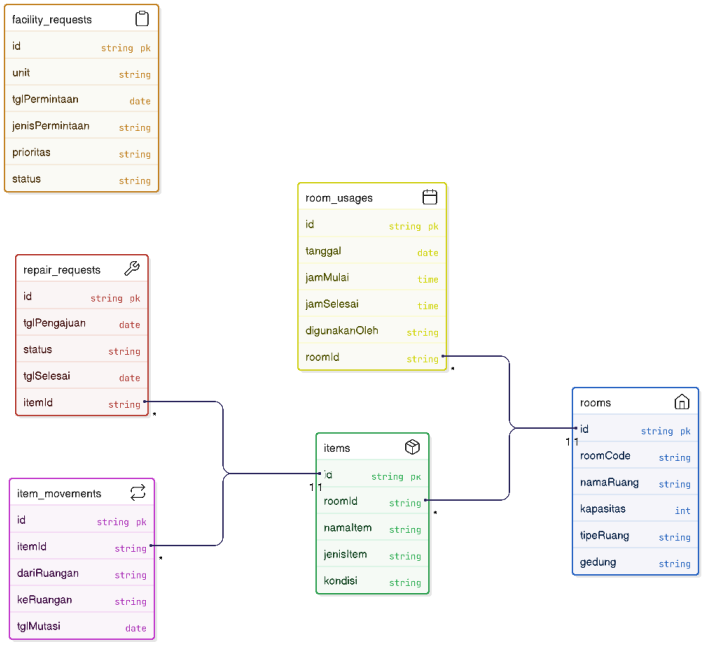
- Dim_Date, Dim_Unit, Dim_Gedung, Dim_RoomType, Dim_Room, Dim_ItemType, Dim_KondisiItem, Dim_Item

Fakta utama:

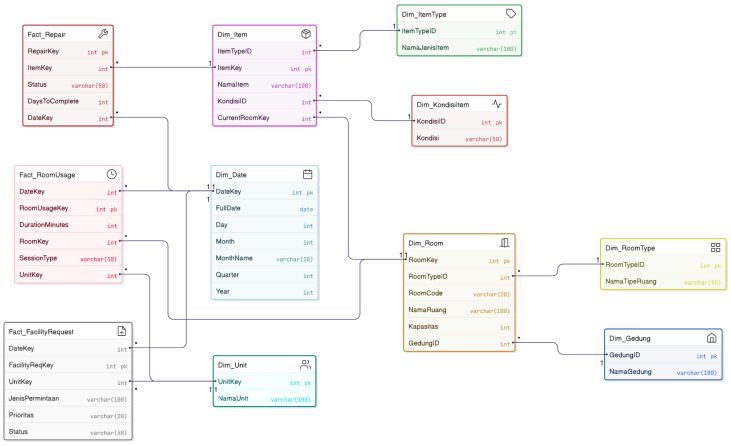
- Fact_RoomUsage, Fact_Repair, Fact_FacilityRequest, Fact_ItemMovement

2.3. Diagram ERD & Star Schema

- ERD menunjukkan relasi antar Dim_ dan Fact_ terbaru.



- Star Schema menempatkan Fact_ di tengah, Dim_ di sekelilingnya untuk query analitis.



2.4. Alur ETL

- Extract: Ambil data dari sistem inventaris dan laporan SARPRAS.
- Transform: Validasi, bersihkan, dan format data.
- Load: Masukkan ke tabel Dim_ dan Fact_ DW.

3. Implementasi

3.1. Setup Database DM_SARPRAS_DW

```
CREATE DATABASE DM_SARPRAS_DW;
GO
USE DM_SARPRAS_DW;
GO
```

3.2. Pembuatan Tabel Dimensi

- Dim_Date

```
-- 1. Dim_Date
CREATE TABLE dbo.Dim_Date (
    DateKey INT PRIMARY KEY,
    FullDate DATE NOT NULL,

    DayNumberOfWeek TINYINT NOT NULL,
    DayName VARCHAR(10) NOT NULL,

    DayNumberOfMonth TINYINT NOT NULL,
    DayNumberOfYear SMALLINT NOT NULL,
    WeekNumberOfYear TINYINT NOT NULL,

    MonthName VARCHAR(20) NOT NULL,
    MonthNumber TINYINT NOT NULL,

    Quarter TINYINT NOT NULL,
    QuarterName VARCHAR(10) NOT NULL,
    Year SMALLINT NOT NULL,

    IsWeekend BIT NOT NULL,
    IsHoliday BIT NOT NULL,
    HolidayName VARCHAR(100) NULL,

    AcademicYear VARCHAR(9) NULL,
    Semester TINYINT NULL
);
GO
```

- Dim_Unit

```
-- 2. Dim_Unit
CREATE TABLE dbo.Dim_Unit (
    UnitKey INT IDENTITY(1,1) PRIMARY KEY,
    NamaUnit VARCHAR(100) NOT NULL
);
GO
```

- Dim_Gedung

```
-- 3. Dim_Gedung
CREATE TABLE dbo.Dim_Gedung (
    GedungID INT IDENTITY(1,1) PRIMARY KEY,
    NamaGedung VARCHAR(100) NOT NULL
);
GO
```

- Dim_RoomType

```
CREATE TABLE dbo.Dim_RoomType (
    RoomTypeID INT IDENTITY(1,1) PRIMARY KEY,
    NamaTipeRuang VARCHAR(100) NOT NULL
);
GO
```

- Dim_Room

```
-- 5. Dim_Room
CREATE TABLE dbo.Dim_Room (
    RoomKey INT IDENTITY(1,1) PRIMARY KEY,
    RoomTypeID INT NOT NULL,
    RoomCode VARCHAR(30) NOT NULL,
    NamaRuang VARCHAR(100) NOT NULL,
    Kapasitas INT NOT NULL,
```

```
    GedungID INT NOT NULL,

    FOREIGN KEY (RoomTypeID) REFERENCES dbo.Dim_RoomType(RoomTypeID),
    FOREIGN KEY (GedungID) REFERENCES dbo.Dim_Gedung(GedungID)
);
GO
```

- Dim_ItemType

```
CREATE TABLE dbo.Dim_ItemType (
    ItemTypeID INT IDENTITY(1,1) PRIMARY KEY,
    NamaJenisItem VARCHAR(100) NOT NULL
);
GO
```

- Dim_KondisiItem

```
-- 7. Dim_KondisiItem
CREATE TABLE dbo.Dim_KondisiItem (
    KondisiID INT IDENTITY(1,1) PRIMARY KEY,
    Kondisi VARCHAR(50) NOT NULL
);
GO
```

- Dim_Item

```
-- 8. Dim_Item
CREATE TABLE dbo.Dim_Item (
    ItemKey INT IDENTITY(1,1) PRIMARY KEY,
    ItemTypeID INT NOT NULL,
    NamaItem VARCHAR(100) NOT NULL,
    KondisiID INT NOT NULL,
    CurrentRoomKey INT NULL,

    FOREIGN KEY (ItemTypeID) REFERENCES dbo.Dim_ItemType(ItemTypeID),
    FOREIGN KEY (KondisiID) REFERENCES dbo.Dim_KondisiItem(KondisiID),
    FOREIGN KEY (CurrentRoomKey) REFERENCES dbo.Dim_Room(RoomKey)
);
GO
```

3.3. Pembuatan Tabel Fakta

- Fact_RoomUsage

```
-- 9. Fact_RoomUsage
CREATE TABLE dbo.Fact_RoomUsage (
    RoomUsageKey BIGINT IDENTITY(1,1) PRIMARY KEY,

    DateKey INT NOT NULL,
    RoomKey INT NOT NULL,
    UnitKey INT NOT NULL,

    DurationMinutes INT NOT NULL,
    SessionType VARCHAR(50) NOT NULL,

    CONSTRAINT FK_RoomUsage_Date
        FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),

    CONSTRAINT FK_RoomUsage_Room
        FOREIGN KEY (RoomKey) REFERENCES dbo.Dim_Room(RoomKey),

    CONSTRAINT FK_RoomUsage_Unit
        FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
```

```
);
GO
```

- Fact_Repair

```
-- 10. Fact_Repair
CREATE TABLE dbo.Fact_Repair (
  RepairKey BIGINT IDENTITY(1,1) PRIMARY KEY,

  ItemKey INT NOT NULL,
  Status VARCHAR(50) NOT NULL,
  DaysToComplete INT NULL,
  DateKey INT NOT NULL,

  CONSTRAINT FK_Repair_Item
    FOREIGN KEY (ItemKey) REFERENCES dbo.Dim_Item(ItemKey),

  CONSTRAINT FK_Repair_Date
    FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey)
);
GO
```

- Fact_FacilityRequest

```
-- 11. Fact_FacilityRequest
CREATE TABLE dbo.Fact_FacilityRequest (
  FacilityReqKey BIGINT IDENTITY(1,1) PRIMARY KEY,

  DateKey INT NOT NULL,
  UnitKey INT NOT NULL,

  JenisPermintaan VARCHAR(200) NOT NULL,
  Prioritas VARCHAR(20) NOT NULL,
  Status VARCHAR(50) NOT NULL,

  CONSTRAINT FK_FacReq_Date
    FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),

  CONSTRAINT FK_FacReq_Unit
    FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
);
GO
```

- Fact_ItemMovement

```
CREATE TABLE dbo.Fact_ItemMovement (
  MovementID BIGINT IDENTITY(1,1) PRIMARY KEY,

  ItemKey INT NOT NULL,
  DariRuangan INT NULL,
  KeRuangan INT NULL,
  TglMutasi DATE NOT NULL,

  CONSTRAINT FK_Movement_Item
    FOREIGN KEY (ItemKey) REFERENCES dbo.Dim_Item(ItemKey),

  CONSTRAINT FK_Movement_Dari
    FOREIGN KEY (DariRuangan) REFERENCES dbo.Dim_Room(RoomKey),

  CONSTRAINT FK_Movement_Ke
    FOREIGN KEY (KeRuangan) REFERENCES dbo.Dim_Room(RoomKey)
);
```

```
GO
```

3.4. Contoh Query Analisis

- **Total durasi penggunaan ruang per unit**:

```
SELECT u.NamaUnit, SUM(f.DurationMinutes) AS TotalDurasi
FROM Fact_RoomUsage f
JOIN Dim_Unit u ON f.UnitKey = u.UnitKey
GROUP BY u.NamaUnit;
```

- **Jumlah perbaikan per kondisi item**:

```
SELECT k.Kondisi, COUNT(r.RepairKey) AS JumlahRepair
FROM Fact_Repair r
JOIN Dim_Item i ON r.ItemKey = i.ItemKey
JOIN Dim_KondisiItem k ON i.KondisiID = k.KondisiID
GROUP BY k.Kondisi;
```

- **Mutasi item per ruangan**:

```
SELECT d.NamaRuang AS RuangAwal, e.NamaRuang AS RuangTujuan, COUNT(m.MovementID) AS JumlahMutasi
FROM Fact_ItemMovement m
LEFT JOIN Dim_Room d ON m.DariRuangan = d.RoomKey
LEFT JOIN Dim_Room e ON m.KeRuangan = e.RoomKey
GROUP BY d.NamaRuang, e.NamaRuang;
```

4. Pengujian & Analisis

4.1. Data Uji

- Data simulasi 10.000+ record mencakup inventaris, ruang, perbaikan, mutasi, dan permintaan fasilitas.

4.2. Hasil Query & Visualisasi

- Diagram batang: total durasi penggunaan ruang per unit.
- Pie chart: jumlah perbaikan per kondisi item.
- Line chart: mutasi item antar ruang sepanjang tahun.

4.3. Analisis Konsistensi & Kualitas Data

- Tidak ada missing key pada tabel Dim_ dan Fact_.
- Relasi foreign key konsisten.
- Query tetap cepat meski jumlah record besar.

5. Penutup

5.1. Kesimpulan

- DW SARPRAS berhasil dirancang dengan dimensi dan fakta lengkap.
- Analisis penggunaan ruang, perbaikan, dan mutasi item dapat dilakukan efisien.
- Sistem siap mendukung pengambilan keputusan berbasis data historis.

5.2. Saran

- Otomatisasi ETL untuk update harian/mingguan.
- Integrasi dengan dashboard visualisasi untuk analisis real-time.
- Perluasan dimensi atau fakta untuk data inventaris tambahan.

MISI 1: DESAIN KONSEPTUAL DAN LOGIKAL

Step 1: Analisis Kebutuhan Bisnis (Business Requirements Analysis)

1. **Identifikasi Stakeholders**

1.1 Pengguna Utama Data Mart

- Kepala Unit Sarana dan Prasarana
- Kepala Bagian Umum
- Staf Administrasi Sarpras
- Unit Perencanaan dan Penganggaran
- Unit Akademik dan Prodi (pengguna ruang kelas/lab)
- Dosen (pemakai ruang kuliah/praktikum)
- Unit Laboratorium dan Studio

1.2 Pengambil Keputusan (Decision Makers)

- Kepala Biro Umum dan Perlengkapan
- Kepala Unit Sarana dan Prasarana
- Unit Perencanaan dan Pengembangan

2. **Analisis Proses Bisnis**

2.1 Proses Bisnis

1. **Manajemen Ruang**
    a. Pendataan ruang, kapasitas, dan tipe ruang
    b. Penjadwalan penggunaan ruang
    c. Pemeriksaan kecukupan kapasitas dengan jumlah mahasiswa

2. **Inventaris Barang**
    a. Pendataan barang per ruangan
    b. Mutasi barang antar ruangan
    c. Pemeriksaan kondisi barang (baik, rusak ringan, rusak berat)

4. **Pemeliharaan dan Perbaikan**
    a. Perawatan fasilitas secara rutin
    b. Penanganan kerusakan ringan
    c. Pencatatan seluruh aktivitas pemeliharaan

5. **Pengajuan Kebutuhan Fasilitas**
    a. Penerimaan permintaan fasilitas dari unit/prodi
    b. Klasifikasi kebutuhan berdasarkan tingkat prioritas
    c. Dokumentasi kebutuhan dan status usulan

4. **Monitoring Utilisasi Ruang dan Fasilitas**
    a. Melihat tingkat pemakaian ruang kelas
    b. Memantau ketersediaan laboratorium

c. Melihat pola penggunaan fasilitas yang sering digunakan

2.2 Key Performance Indicators (KPI)

2.2.1 KPI Tingkat Institusi

- Persentase ketersediaan ruang kelas
- Tingkat kelayakan fasilitas kampus
- Tingkat pemanfaatan ruang (room utilization rate)

2.2.2 KPI Tingkat Unit / Program Studi

- Rata-rata penggunaan ruang praktikum
- Kesesuaian kapasitas ruang dengan jumlah mahasiswa
- Persentase penyelesaian laporan kerusakan

2.2.3 KPI Operasional Sarpras

- Waktu respon penanganan perbaikan
- Jumlah fasilitas rusak per bulan
- Jumlah permintaan fasilitas baru per unit
- Jumlah barang yang mengalami mutasi

2.3 Kebutuhan Analitik

2.3.1 Pertanyaan Analitik yang Akan Dijawab

- Ruangan mana yang paling sering digunakan?
- Bagaimana tren kondisi barang dari bulan ke bulan?
- Apakah kapasitas ruang sudah sesuai dengan jumlah mahasiswa?
- Berapa banyak laporan perbaikan yang diselesaikan tepat waktu?
- Barang apa saja yang paling sering mengalami kerusakan?

2.3.2 Laporan yang Diperlukan

- Dashboard pemanfaatan ruang
- Laporan kondisi fasilitas
- Laporan perawatan dan perbaikan
- Laporan permintaan fasilitas baru
- Laporan mutasi barang

Step 2: Identifikasi Sumber Data (Data Source Identification)

1. Sumber Data Utama

- Sistem Manajemen Ruang (internal kampus)
- Data Inventaris Sarpras (Excel/Database internal)
- Formulir pengajuan perbaikan

- Formulir permintaan fasilitas baru
- Dokumen mutasi barang

2. Data Source Analysis

Tabel 2.1

Sumber Data	**Tipe**	**Tabel Utama**	**Volume**	**Frekuensi Update**
Sistem Ruang	SQL Server	Rooms, RoomUsage	~10k	Harian
Inventaris Barang	Excel/DB	Items, ItemStatus	~50k	Mingguan
Pengajuan Perbaikan	Form Online	RepairRequest	~5k	Harian
Permintaan Fasilitas	Form Online	FacilityRequest	~2k	Mingguan
Mutasi Barang	Excel	ItemMovement	kecil	Tidak tentu

Pada tabel 2.1 sistem data terdiri dari lima sumber utama yang digunakan untuk mendukung pengelolaan sarana dan prasarana. Sistem Ruang menggunakan SQL Server sebagai basis data utama, dengan tabel Rooms dan RoomUsage yang menyimpan sekitar 10.000 data dan diperbarui setiap hari untuk memastikan informasi penggunaan ruangan selalu terbaru. Selanjutnya, Inventaris Barang tersimpan dalam format Excel maupun database, mencakup tabel Items dan ItemStatus dengan volume data cukup besar, yaitu sekitar 50.000 baris, dan diperbarui setiap minggu. Untuk Pengajuan Perbaikan, data berasal dari formulir online dan disimpan dalam tabel RepairRequest dengan jumlah sekitar 5.000 data yang diperbarui secara harian karena sifatnya yang operasional dan membutuhkan respons cepat. Permintaan Fasilitas Baru juga dikumpulkan melalui formulir online dan dicatat dalam tabel FacilityRequest, dengan volume sekitar 2.000 data dan diperbarui mingguan. Terakhir, Mutasi Barang dicatat melalui file Excel pada tabel ItemMovement, dengan jumlah data relatif kecil dan frekuensi yang tidak menentu, tergantung ada atau tidaknya perpindahan barang.

3. Data Profiling

- Periksa NULL pada RoomCode, ConditionStatus
- Pastikan kapasitas ruang adalah angka positif
- Cek duplikasi pada ItemID dan RoomUsageID
- Konsistensi tanggal pemeliharaan
- Validasi relasi: item.room_id → rooms.room_id

Step 3: Desain Konseptual – ERD

1. Entitas Utama

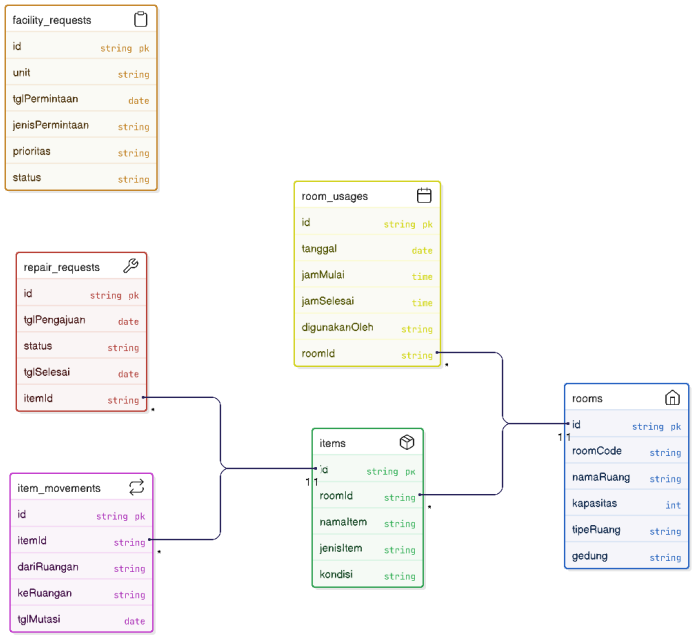
- Room (RoomID, RoomCode, NamaRuang, Kapasitas, TipeRuang, Gedung)
- RoomUsage (UsageID, RoomID, Tanggal, JamMulai, JamSelesai, DigunakanOleh)
- Item (ItemID, NamaItem, JenisItem, Kondisi, RoomID)

- RepairRequest (RepairID, ItemID, TglPengajuan, Status, TglSelesai)
- FacilityRequest (ReqID, Unit, TglPermintaan, JenisPermintaan, Prioritas, Status)
- ItemMovement (MovementID, ItemID, DariRuangan, KeRuangan, TglMutasi)

2. Definisi Relationships

- Room 1–N RoomUsage
- Room 1–N Item
- Item 1–N RepairRequest
- Item 1–N ItemMovement
- FacilityRequest berdiri sendiri (tidak wajib terhubung dengan ruang/barang)

3. Gambar ERD



Gambar 3.1

3.1 Deskripsi setiap entitas dan relationship

Pada gambar 3.1 diagram menggambarkan hubungan antar tabel dalam sistem manajemen fasilitas yang mengatur ruangan, barang, serta aktivitas yang berkaitan dengannya. Tabel rooms berisi informasi dasar mengenai setiap ruangan, seperti kode, nama, kapasitas, tipe ruangan, dan gedung tempat ruangan berada. Setiap ruangan dapat memiliki banyak barang, yang datanya disimpan dalam tabel items. Tabel ini mencatat nama barang, jenis, kondisi, serta ruangan tempat barang tersebut berada. Barang yang tersimpan di ruangan dapat mengalami perpindahan, dan riwayatnya dicatat dalam tabel item_movements, yang menunjukkan barang apa yang dipindahkan, berasal dari ruangan mana, dipindahkan ke ruangan mana, serta tanggal mutasinya. Selain itu, barang juga dapat diajukan untuk perbaikan; proses tersebut direkam dalam tabel repair_requests, yang memuat tanggal pengajuan, status perbaikan, tanggal selesai, dan barang yang mengalami kerusakan.

Di sisi lain, penggunaan ruangan dicatat dalam tabel room_usages, yang berisi informasi mengenai tanggal pemakaian, jam mulai, jam selesai, serta siapa pengguna ruangan tersebut. Setiap catatan pemakaian terhubung ke satu ruangan dalam tabel rooms. Terakhir, terdapat tabel facility_requests yang berdiri sendiri dan digunakan untuk menampung permintaan fasilitas dari unit pengaju, seperti jenis permintaan, prioritas, tanggal permintaan, dan statusnya.

3.2 Dokumen asumsi dan business rules

3.2.1 Dokumen asumsi

1. Setiap ruangan memiliki informasi lengkap seperti kode ruangan, nama ruangan, kapasitas, tipe ruangan, dan gedung yang bersifat tetap kecuali ada perubahan dari pihak pengelola fasilitas.
2. Setiap barang (item) pasti berada dalam satu ruangan tertentu, sehingga *roomId* pada tabel items selalu valid.
3. Setiap barang hanya memiliki satu kondisi pada satu waktu, misalnya “Baik”, “Rusak Ringan”, atau “Rusak Berat”.
4. Setiap perbaikan barang (repair_requests) yang tercatat pasti dilakukan terhadap barang yang sudah terdaftar di tabel items.
5. Setiap mutasi barang (item_movements) diasumsikan dicatat pada tanggal terjadinya perpindahan (tglMutasi).
6. Data penggunaan ruangan (room_usages) dianggap akurat, sehingga waktu mulai dan waktu selesai sudah sesuai kenyataan.
7. Setiap penggunaan ruangan hanya untuk satu kegiatan pada satu waktu, sehingga tidak ada dua kegiatan berbeda yang menggunakan ruangan sama di jam yang sama.
8. Permintaan fasilitas (facility_requests) diajukan oleh unit resmi dan dicatat pada tanggal pengajuan yang valid.
9. Semua tanggal dan waktu yang dicatat telah mengikuti format standar sistem, misalnya YYYY-MM-DD untuk tanggal.
10. ERD menggambarkan sistem operasional, sehingga setiap tabel menyimpan catatan aktual kegiatan, bukan data historis analisis.

3.2.1 Business rules

A. Aturan Barang dan Mutasi (Items & Item Movements)

1. Setiap item harus memiliki roomId yang valid, artinya barang tidak boleh tercatat tanpa ruangan.
2. Item hanya dapat dipindahkan melalui proses mutasi yang dicatat di tabel item_movements.
3. Pada saat mutasi terjadi, kondisi barang tetap merujuk pada kondisi terbaru yang tercatat pada tabel items.
4. Setiap mutasi harus mencatat ruangan asal dan ruangan tujuan, keduanya harus berupa ruangan yang terdaftar.

B. Aturan Perbaikan Barang (Repair Requests)

1. Setiap pengajuan perbaikan harus memiliki itemId yang valid, menunjukkan barang tersebut memang ada di sistem.
2. Status perbaikan harus mengikuti daftar status standar, seperti: Pending, Diproses, atau Selesai.
3. Tanggal selesai hanya boleh diisi jika status sudah “Selesai”.

4. Barang yang sedang diperbaiki tetap terikat pada ruangan terakhirnya, kecuali dilakukan mutasi setelah perbaikan.

C. Aturan Penggunaan Ruangan (Room Usages)

1. Setiap penggunaan ruangan harus mengacu pada ruangan yang ada di tabel rooms.
2. Jam mulai harus lebih awal dari pada jam selesai, dan tidak boleh sama.
3. Ruangan tidak boleh digunakan oleh dua pihak berbeda dalam waktu yang bertumpuk.
4. Setiap pemakaian ruangan harus mencantumkan pihak pengguna melalui field digunakan Oleh.

D. Aturan Permintaan Fasilitas (Facility Requests)

1. Setiap permintaan fasilitas harus mencantumkan unit dan jenis permintaan, keduanya wajib diisi.
2. Prioritas harus mengikuti format standar, misalnya: Low, Medium, atau High.
3. Status permintaan harus mengikuti aturan, seperti: Pending, Diproses, atau Selesai.
4. Tanggal permintaan tidak boleh kosong, dan menjadi acuan proses pelayanan fasilitas.

E. Aturan Integritas Relasi

1. Semua foreign key harus memiliki referensi yang valid, seperti itemId → items.id dan roomId → rooms.id.
2. Data tidak boleh dihapus jika masih menjadi referensi oleh tabel lain (misalnya barang yang punya riwayat perbaikan tidak bisa dihapus begitu saja).
3. Tabel items menjadi pusat referensi bagi mutasi dan perbaikan, sehingga setiap perubahan harus konsisten.
4. Setiap catatan harus mengikuti format tipe data yang ditentukan (misalnya time, date, string, int).

Step 4: Desain Logikal – Dimensional Model (Star Schema)

1. Tabel Fakta

a. Fact_RoomUsage

- RoomUsageKey
- DateKey
- RoomKey
- UnitKey
- DurationMinutes
- JenisSesi

b. Fact_Repair

- RepairKey
- DateKey
- ItemKey
- Status

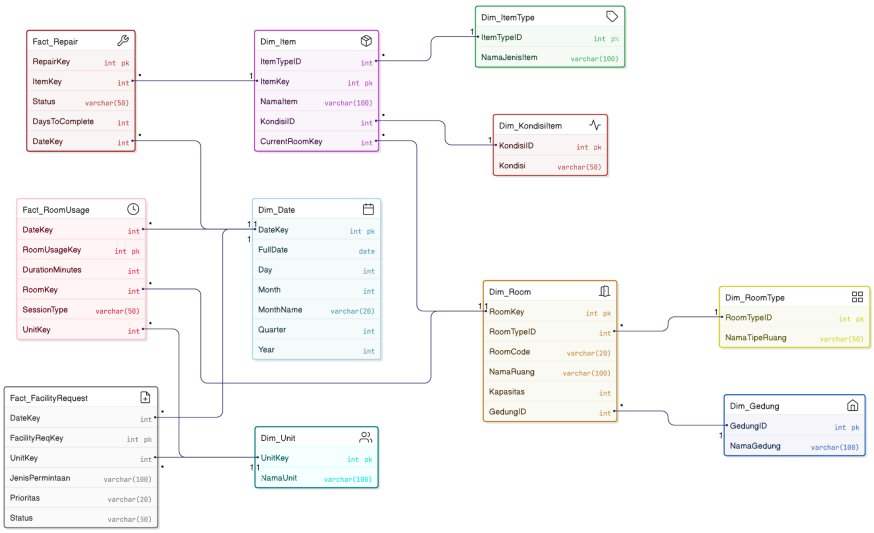
- LamaPenyelesaian (hari)

c. **Fact_FacilityRequest**
   - FacilityReqKey
   - DateKey
   - UnitKey
   - JenisPermintaan
   - Prioritas
   - Status

2. Tabel Dimensi

- Dim_Room — info ruang
- Dim_Item — info barang/fasilitas
- Dim_Unit — info unit/prodi
- Dim_Date — kalender lengkap

3. Desain Star/Snowflake Schema



Gambar 4.1

Pada gambar 4.1 diagram menunjukkan struktur data berbentuk data warehouse dengan model star schema, yang digunakan untuk menganalisis aktivitas fasilitas, pemakaian ruangan, permintaan fasilitas, dan perbaikan barang. Tabel-tabel berwarna merah merupakan tabel fakta yang menyimpan catatan peristiwa atau aktivitas, sedangkan tabel berwarna lain adalah tabel dimensi yang berisi informasi detail untuk menjelaskan setiap aktivitas. Tabel Fact_Repair menyimpan data mengenai perbaikan barang, seperti barang yang diperbaiki, status, lama waktu penyelesaian, dan tanggalnya. Detail barang tersebut berasal dari tabel Dim_Item, yang berisi informasi lengkap tentang barang, termasuk jenis barang yang diambil dari Dim_ItemType, kondisi barang dari Dim_KondisiItem, serta lokasi ruangannya melalui CurrentRoomKey yang terhubung ke Dim_Room.

Untuk aktivitas penggunaan ruangan, data tersebut dicatat dalam Fact_RoomUsage, yang memuat informasi seperti tanggal penggunaan, lama penggunaan, jenis sesi, ruangan yang digunakan, dan unit yang menggunakan ruangan. Informasi ruangan diambil dari tabel Dim_Room, yang memuat nama ruangan, tipe

ruangan, kapasitas, serta gedungnya. Tipe ruangan dijelaskan lebih lanjut di Dim_RoomType, sedangkan nama gedungnya berasal dari Dim_Gedung. Aktivitas lain berupa permintaan fasilitas tercatat dalam Fact_FacilityRequest, yang menyimpan data seperti tanggal permintaan, unit pengaju, jenis permintaan, prioritas, dan status. Unit pengaju diambil dari tabel Dim_Unit. Semua aktivitas yang melibatkan tanggal terhubung ke tabel Dim_Date, yang menyimpan detail waktu seperti hari, bulan, tahun, nama bulan, dan kuartal.

Step 5: Data Dictionary

1. **Tabel Analisis Sumber Data**

Tabel 5.1

Table	**Column**	**Data Type**	**PK/FK**	**Description**	**Business Rule**
Dim_Room	RoomKey	INT	PK	Surrogate key ruangan	Auto-increment
Dim_Room	RoomTypeID	INT	FK	Jenis/tipe ruang	Harus ada di Dim_RoomType
Dim_Room	RoomCode	VARCHAR(20)		Kode ruang	Unique, NOT NULL
Dim_Room	NamaRuang	VARCHAR(100)		Nama ruangan	NOT NULL
Dim_Room	Kapasitas	INT		Kapasitas maksimum	> 0
Dim_Room	GedungID	INT	FK	Lokasi gedung	Harus ada di Dim_Gedung
Dim_Gedung	GedungID	INT	PK	ID gedung	Auto-increment
Dim_Gedung	NamaGedung	VARCHAR(100)		Nama gedung	NOT NULL
Dim_RoomType	RoomTypeID	INT	PK	ID tipe ruangan	Auto-increment
Dim_RoomType	NamaTipeRuang	VARCHAR(50)		Nama tipe ruang	NOT NULL
Dim_Item	ItemKey	INT	PK	Surrogate key item	Auto-increment
Dim_Item	ItemTypeID	INT	FK	Jenis item	Harus ada di Dim_ItemType

Table	Column	Data Type	PK/FK	Description	Business Rule
Dim_Item	NamaItem	VARCHAR(100)		Nama barang	NOT NULL
Dim_Item	KondisiID	INT	FK	Status kondisi barang	Harus ada di Dim_KondisiItem
Dim_Item	CurrentRoomKey	INT	FK	Lokasi ruangan saat ini	Harus ada di Dim_Room
Dim_ItemType	ItemTypeID	INT	PK	ID jenis item	Auto-increment
Dim_ItemType	NamaJenisItem	VARCHAR(100)		Nama kategori item	NOT NULL
Dim_KondisiItem	KondisiID	INT	PK	ID kondisi	Auto-increment
Dim_KondisiItem	Kondisi	VARCHAR(50)		Nama kondisi barang	(Baik/Rusak/Rusak Berat)
Dim_Unit	UnitKey	INT	PK	Surrogate key unit	Auto-increment
Dim_Unit	NamaUnit	VARCHAR(100)		Nama unit/prodi	NOT NULL
Dim_Date	DateKey	INT	PK	YYYYMMDD	Derived from FullDate
Dim_Date	FullDate	DATE		Tanggal lengkap	Valid date
Dim_Date	Day	INT		Hari	1–31
Dim_Date	Month	INT		Bulan	1–12
Dim_Date	MonthName	VARCHAR(20)		Nama bulan	Auto-generate
Dim_Date	Quarter	INT		Kuartal	1–4
Dim_Date	Year	INT		Tahun	4-digit
Fact_RoomUsage	RoomUsageKey	INT	PK	Surrogate key	Auto-increment

Table	Column	Data Type	PK/FK	Description	Business Rule
Fact_RoomUsage	DateKey	INT	FK	Tanggal penggunaan	Harus ada di Dim_Date
Fact_RoomUsage	RoomKey	INT	FK	Ruang yang dipakai	Harus ada di Dim_Room
Fact_RoomUsage	UnitKey	INT	FK	Unit pengguna	Harus ada di Dim_Unit
Fact_RoomUsage	DurationMinutes	INT		Lama penggunaan	≥ 0
Fact_RoomUsage	SessionType	VARCHAR(50)		Jenis pemakaian	Kuliah / Praktikum / Rapat
Fact_Repair	RepairKey	INT	PK	Surrogate key	Auto-increment
Fact_Repair	ItemKey	INT	FK	Barang yang diperbaiki	Harus ada di Dim_Item
Fact_Repair	Status	VARCHAR(50)		Status perbaikan	Open / Process / Done
Fact_Repair	DaysToComplete	INT		Lama penyelesaian	≥ 0
Fact_Repair	DateKey	INT	FK	Tanggal laporan	Harus ada di Dim_Date
Fact_FacilityRequest	FacilityReqKey	INT	PK	Surrogate key	Auto-increment
Fact_FacilityRequest	DateKey	INT	FK	Tanggal permintaan	Harus ada di Dim_Date
Fact_FacilityRequest	UnitKey	INT	FK	Unit pemohon	Harus ada di Dim_Unit
Fact_FacilityRequest	JenisPermintaan	VARCHAR(100)		Tipe fasilitas	NOT NULL
Fact_FacilityRequest	Prioritas	VARCHAR(20)		Prioritas	Rendah/Sedang/Tinggi
Fact_FacilityRequest	Status	VARCHAR(50)		Status usulan	Diajukan/Proses/Selesai
ItemMovement	MovementID	INT	PK	Surrogate key	Auto-increment

Table	Column	Data Type	PK/FK	Description	Business Rule
ItemMovement	ItemKey	INT	FK	Barang yang dipindah	Harus ada di Dim_Item
ItemMovement	DariRuangan	INT	FK	Ruang asal	Harus ada di Dim_Room
ItemMovement	KeRuangan	INT	FK	Ruang tujuan	Harus ada di Dim_Room
ItemMovement	TglMutasi	DATE		Tanggal perpindahan	Valid date

Pada tabel 5.1 Tabel Dim_Room digunakan untuk menyimpan informasi detail mengenai setiap ruangan yang ada di kampus. Setiap ruangan memiliki RoomKey sebagai identitas unik yang dibuat otomatis. Selain itu, ruangan dikategorikan berdasarkan RoomTypeID yang merujuk ke tabel tipe ruangan. Ruangan juga memiliki RoomCode yang wajib unik dan NamaRuang yang wajib diisi, serta Kapasitas yang menunjukkan jumlah maksimum orang yang dapat ditampung. Setiap ruangan pasti berada dalam sebuah gedung tertentu, sehingga menyertakan GedungID sebagai referensi ke tabel gedung.

Tabel Dim_Gedung berisi daftar seluruh gedung di kampus. Setiap gedung punya GedungID sebagai kunci utama dan NamaGedung sebagai penanda nama gedung tersebut. Sementara itu, tabel Dim_RoomType menyimpan informasi mengenai jenis atau kategori ruangan, seperti ruang kelas, laboratorium, atau studio, lengkap dengan RoomTypeID sebagai kunci utama dan NamaTipeRuang sebagai nama tipenya.

Tabel Dim_Item digunakan untuk mencatat semua barang inventaris yang ada, mulai dari proyektor hingga kursi dan meja. Setiap barang memiliki ItemKey sebagai kunci utama, NamaItem sebagai nama barang, serta ItemTypeID yang menghubungkannya ke tabel jenis barang. Barang juga memiliki KondisiID yang menunjukkan kondisinya (baik, rusak, atau rusak berat) dan CurrentRoomKey yang menunjukkan ruangan tempat barang berada saat ini. Informasi terkait jenis barang tersimpan di tabel Dim_ItemType, yang berisi ItemTypeID dan NamaJenisItem seperti “Elektronik” atau “Furnitur”. Sementara itu, kondisi barang disimpan dalam tabel Dim_KondisiItem, yang mencakup daftar kondisi standar seperti “Baik”, “Rusak”, dan “Rusak Berat”.

Tabel Dim_Unit mencatat daftar unit, program studi, ataupun lembaga yang menggunakan fasilitas kampus. Setiap unit memiliki UnitKey dan NamaUnit yang wajib diisi. Untuk kebutuhan tanggal, seluruh data menggunakan tabel Dim_Date sebagai referensi kalender lengkap. Tabel ini menyimpan informasi tanggal secara terstruktur, mulai dari DateKey (dalam format YYYYMMDD), FullDate, hingga atribut tanggal seperti hari, bulan, kuartal, dan tahun.

Pada bagian transaksi, tabel Fact_RoomUsage mencatat setiap kejadian penggunaan ruangan. Informasi yang tersimpan mencakup tanggal penggunaan, ruangan yang digunakan, unit yang menggunakan, durasi penggunaan dalam menit, dan jenis sesi seperti kuliah, praktikum, atau rapat. Selanjutnya, tabel Fact_Repair menyimpan catatan perbaikan barang, termasuk barang yang

diperbaiki, status perbaikan (open, process, done), lama penyelesaian, dan tanggal laporan. Tabel Fact_FacilityRequest mencatat permintaan fasilitas dari berbagai unit, mencakup jenis permintaan, prioritas, status, serta unit yang mengajukan permintaan tersebut.

Terakhir, tabel ItemMovement digunakan untuk mencatat proses pemindahan barang antar-ruangan. Setiap catatan perpindahan memuat barang apa yang dipindahkan, ruangan asal dan tujuan, serta tanggal mutasinya. Tabel ini membantu melacak riwayat pergerakan barang sehingga inventaris dapat dipantau secara akurat.

MISI 2: DESAIN FISIKAL DAN DEVELOPMENT

Step 1: Physical Database Design

Tujuan: Mengimplementasikan dimensional model ke SQL Server

Aktivitas:

1. Database Setup

```
CREATE DATABASE DM_SARPRAS_DW;
GO
USE DM_SARPRAS_DW;
GO
```

2. Create Dimension Tables
   - Dim_Date

```
-- 1. Dim_Date
CREATE TABLE dbo.Dim_Date (
    DateKey INT PRIMARY KEY,
    FullDate DATE NOT NULL,

    DayNumberOfWeek TINYINT NOT NULL,
    DayName VARCHAR(10) NOT NULL,

    DayNumberOfMonth TINYINT NOT NULL,
    DayNumberOfYear SMALLINT NOT NULL,
    WeekNumberOfYear TINYINT NOT NULL,

    MonthName VARCHAR(20) NOT NULL,
    MonthNumber TINYINT NOT NULL,

    Quarter TINYINT NOT NULL,
    QuarterName VARCHAR(10) NOT NULL,
    Year SMALLINT NOT NULL,

    IsWeekend BIT NOT NULL,
    IsHoliday BIT NOT NULL,
    HolidayName VARCHAR(100) NULL,

    AcademicYear VARCHAR(9) NULL,
    Semester TINYINT NULL
);
GO
```

   - Dim_Unit

```
-- 2. Dim_Unit
CREATE TABLE dbo.Dim_Unit (
    UnitKey INT IDENTITY(1,1) PRIMARY KEY,
    NamaUnit VARCHAR(100) NOT NULL
);
GO
```

   - Dim_Gedung

```
-- 3. Dim_Gedung
CREATE TABLE dbo.Dim_Gedung (
    GedungID INT IDENTITY(1,1) PRIMARY KEY,
    NamaGedung VARCHAR(100) NOT NULL
);
GO
```

- Dim_RoomType

```
CREATE TABLE dbo.Dim_RoomType (
   RoomTypeID INT IDENTITY(1,1) PRIMARY KEY,
   NamaTipeRuang VARCHAR(100) NOT NULL
);
GO
```

- Dim_Room

```
-- 5. Dim_Room
CREATE TABLE dbo.Dim_Room (
   RoomKey INT IDENTITY(1,1) PRIMARY KEY,
   RoomTypeID INT NOT NULL,
   RoomCode VARCHAR(30) NOT NULL,
   NamaRuang VARCHAR(100) NOT NULL,
   Kapasitas INT NOT NULL,
   GedungID INT NOT NULL,

   FOREIGN KEY (RoomTypeID) REFERENCES dbo.Dim_RoomType(RoomTypeID),
   FOREIGN KEY (GedungID) REFERENCES dbo.Dim_Gedung(GedungID)
);
GO
```

- Dim_ItemType

```
CREATE TABLE dbo.Dim_ItemType (
   ItemTypeID INT IDENTITY(1,1) PRIMARY KEY,
   NamaJenisItem VARCHAR(100) NOT NULL
);
GO
```

- Dim_KondisiItem

```
-- 7. Dim_KondisiItem
CREATE TABLE dbo.Dim_KondisiItem (
   KondisiID INT IDENTITY(1,1) PRIMARY KEY,
   Kondisi VARCHAR(50) NOT NULL
);
GO
```

- Dim_Item

```
-- 8. Dim_Item
CREATE TABLE dbo.Dim_Item (
   ItemKey INT IDENTITY(1,1) PRIMARY KEY,
   ItemTypeID INT NOT NULL,
   NamaItem VARCHAR(100) NOT NULL,
   KondisiID INT NOT NULL,
   CurrentRoomKey INT NULL,

   FOREIGN KEY (ItemTypeID) REFERENCES dbo.Dim_ItemType(ItemTypeID),
   FOREIGN KEY (KondisiID) REFERENCES dbo.Dim_KondisiItem(KondisiID),
   FOREIGN KEY (CurrentRoomKey) REFERENCES dbo.Dim_Room(RoomKey)
);
GO
```

3. Create Fact Tables
    - Fact_RoomUsage

```
-- 9. Fact_RoomUsage
CREATE TABLE dbo.Fact_RoomUsage (
   RoomUsageKey BIGINT IDENTITY(1,1) PRIMARY KEY,
```

```
    DateKey INT NOT NULL,
    RoomKey INT NOT NULL,
    UnitKey INT NOT NULL,

    DurationMinutes INT NOT NULL,
    SessionType VARCHAR(50) NOT NULL,

    CONSTRAINT FK_RoomUsage_Date
      FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),

    CONSTRAINT FK_RoomUsage_Room
      FOREIGN KEY (RoomKey) REFERENCES dbo.Dim_Room(RoomKey),

    CONSTRAINT FK_RoomUsage_Unit
      FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
);
GO
```

- Fact_Repair

```
-- 10. Fact_Repair
CREATE TABLE dbo.Fact_Repair (
    RepairKey BIGINT IDENTITY(1,1) PRIMARY KEY,

    ItemKey INT NOT NULL,
    Status VARCHAR(50) NOT NULL,
    DaysToComplete INT NULL,
    DateKey INT NOT NULL,

    CONSTRAINT FK_Repair_Item
      FOREIGN KEY (ItemKey) REFERENCES dbo.Dim_Item(ItemKey),

    CONSTRAINT FK_Repair_Date
      FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey)
);
GO
```

- Fact_FacilityRequest

```
-- 11. Fact_FacilityRequest
CREATE TABLE dbo.Fact_FacilityRequest (
    FacilityReqKey BIGINT IDENTITY(1,1) PRIMARY KEY,

    DateKey INT NOT NULL,
    UnitKey INT NOT NULL,

    JenisPermintaan VARCHAR(200) NOT NULL,
    Prioritas VARCHAR(20) NOT NULL,
    Status VARCHAR(50) NOT NULL,

    CONSTRAINT FK_FacReq_Date
      FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),

    CONSTRAINT FK_FacReq_Unit
      FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
);
GO
```

- Fact_ItemMovement

```
CREATE TABLE dbo.Fact_ItemMovement (
    MovementID BIGINT IDENTITY(1,1) PRIMARY KEY,
```

```
    ItemKey INT NOT NULL,
    DariRuangan INT NULL,
    KeRuangan INT NULL,
    TglMutasi DATE NOT NULL,

    CONSTRAINT FK_Movement_Item
      FOREIGN KEY (ItemKey) REFERENCES dbo.Dim_Item(ItemKey),

    CONSTRAINT FK_Movement_Dari
      FOREIGN KEY (DariRuangan) REFERENCES dbo.Dim_Room(RoomKey),

    CONSTRAINT FK_Movement_Ke
      FOREIGN KEY (KeRuangan) REFERENCES dbo.Dim_Room(RoomKey)
);
GO
```

Step 2: Indexing Strategy

Tujuan: Mengoptimalkan query performance dengan indexing yang tepat

Aktivitas:

1. Non-Clustered Indexes
    - Fact_RoomUsage

```
CREATE NONCLUSTERED INDEX IX_Fact_Repair_Covering
ON dbo.Fact_Repair(ItemKey, DateKey)
INCLUDE (DaysToComplete, Status);
GO
CREATE NONCLUSTERED INDEX IX_Fact_RoomUsage_Covering
ON dbo.Fact_RoomUsage(DateKey, RoomKey)
INCLUDE (UnitKey, DurationMinutes, SessionType);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Repair_Item
ON dbo.Fact_Repair(ItemKey)
INCLUDE (DaysToComplete, Status);
GO

CREATE NONCLUSTERED INDEX IX_Fact_Repair_Date
ON dbo.Fact_Repair(DateKey)
INCLUDE (DaysToComplete, Status);
GO

CREATE NONCLUSTERED INDEX IX_Fact_FacilityRequest_Covering
ON dbo.Fact_FacilityRequest(DateKey, UnitKey)
INCLUDE (JenisPermintaan, Prioritas, Status);
GO

CREATE NONCLUSTERED INDEX IX_Fact_ItemMovement_Item
ON dbo.Fact_ItemMovement(ItemKey)
INCLUDE (DariRuangan, KeRuangan, TglMutasi);
GO

CREATE NONCLUSTERED INDEX IX_Fact_ItemMovement_Rooms
ON dbo.Fact_ItemMovement(DariRuangan, KeRuangan)
INCLUDE (TglMutasi, ItemKey);
GO
```

2. Columnstore Index

```
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_RoomUsage
```

```
ON dbo.Fact_RoomUsage
(
   DateKey, RoomKey, UnitKey,
   DurationMinutes, SessionType
);
GO
CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_Repair
ON dbo.Fact_Repair
(
   ItemKey, DateKey,
   DaysToComplete, Status
);
GO

CREATE NONCLUSTERED COLUMNSTORE INDEX NCCIX_Fact_FacilityRequest
ON dbo.Fact_FacilityRequest
(
   DateKey, UnitKey,
   JenisPermintaan, Prioritas, Status
);
GO
```

	TableName	IndexName	IndexType	is_primary_key	is_unique
1	Dim_Date	PK__Dim_Date__40DF45E3BBCE5518	CLUSTERED	1	1
2	Dim_Gedung	PK__Dim_Gedu__BD03D5350BFAD732	CLUSTERED	1	1
3	Dim_Item	PK__Dim_Item__136EF54315BFAF7B	CLUSTERED	1	1
4	Dim_ItemType	PK__Dim_Item__F51540DBDD1E99AA	CLUSTERED	1	1
5	Dim_Kondisiltem	PK__Dim_Kond__817F8D838F40EC5A	CLUSTERED	1	1
6	Dim_Room	PK__Dim_Room__A238A347F3A48BF3	CLUSTERED	1	1
7	Dim_RoomType	PK__Dim_Room__BCC8961145041E4E	CLUSTERED	1	1
8	Dim_Unit	PK__Dim_Unit__1C396085F3E15616	CLUSTERED	1	1
9	Fact_FacilityRequest	IX_Fact_FacilityRequest_Covering	NONCLUSTERED	0	0
10	Fact_FacilityRequest	NCCIX_Fact_FacilityRequest	NONCLUSTERED COLUMNSTORE	0	0
11	Fact_FacilityRequest	PK__Fact_Fac__FCA1D2EA1AFEAFEE	CLUSTERED	1	1
12	Fact_ItemMovement	IX_Fact_ItemMovement_Item	NONCLUSTERED	0	0
13	Fact_ItemMovement	IX_Fact_ItemMovement_Rooms	NONCLUSTERED	0	0
14	Fact_ItemMovement	PK__Fact_Ite__D18224663DE45AD3	CLUSTERED	1	1
15	Fact_Repair	IX_Fact_Repair_Covering	NONCLUSTERED	0	0
16	Fact_Repair	IX_Fact_Repair_Date	NONCLUSTERED	0	0
17	Fact_Repair	IX_Fact_Repair_Item	NONCLUSTERED	0	0
18	Fact_Repair	NCCIX_Fact_Repair	NONCLUSTERED COLUMNSTORE	0	0
19	Fact_Repair	PK__Fact_Rep__7ABCF6705D262C0B	CLUSTERED	1	1
20	Fact_RoomUsage	IX_Fact_RoomUsage_Covering	NONCLUSTERED	0	0
21	Fact_RoomUsage	NCCIX_Fact_RoomUsage	NONCLUSTERED COLUMNSTORE	0	0
22	Fact_RoomUsage	PK__Fact_Roo__CA1B81C80E328457	CLUSTERED	1	1

Step 3: Partitioning Strategy

Tujuan: Meningkatkan manageability dan performance untuk large tables

Aktivitas:

```
IF OBJECT_ID('dbo.Fact_RoomUsage_Partitioned', 'U') IS NOT NULL DROP TABLE dbo.Fact_RoomUsage_Partitioned;
IF OBJECT_ID('dbo.Fact_FacilityRequest_Partitioned', 'U') IS NOT NULL DROP TABLE dbo.Fact_FacilityRequest_Partitioned;
IF OBJECT_ID('dbo.Fact_ItemMovement_Partitioned', 'U') IS NOT NULL DROP TABLE dbo.Fact_ItemMovement_Partitioned;
GO

IF EXISTS (SELECT * FROM sys.partition_schemes WHERE name = 'PS_DateKey_Year')
   DROP PARTITION SCHEME PS_DateKey_Year;
GO

IF EXISTS (SELECT * FROM sys.partition_functions WHERE name = 'PF_DateKey_Year')
   DROP PARTITION FUNCTION PF_DateKey_Year;
GO

CREATE PARTITION FUNCTION PF_DateKey_Year (INT)
AS RANGE RIGHT FOR VALUES (2020, 2021, 2022, 2023, 2024, 2025, 2026);
GO


CREATE PARTITION SCHEME PS_DateKey_Year
AS PARTITION PF_DateKey_Year
```

```
ALL TO ([PRIMARY]);
GO

CREATE TABLE dbo.Fact_RoomUsage_Partitioned
(
   RoomUsageKey BIGINT NOT NULL,
   DateKey INT NOT NULL,
   RoomKey INT NOT NULL,
   UnitKey INT NOT NULL,
   DurationMinutes INT NOT NULL,
   SessionType VARCHAR(50) NOT NULL,

   CONSTRAINT FK_RoomUsageP_Date
      FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),
   CONSTRAINT FK_RoomUsageP_Room
      FOREIGN KEY (RoomKey) REFERENCES dbo.Dim_Room(RoomKey),
   CONSTRAINT FK_RoomUsageP_Unit
      FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
)
ON PS_DateKey_Year(DateKey);
GO

ALTER TABLE dbo.Fact_RoomUsage_Partitioned
ADD CONSTRAINT PK_Fact_RoomUsage_Partitioned
PRIMARY KEY CLUSTERED (RoomUsageKey, DateKey)
ON PS_DateKey_Year(DateKey);
GO

CREATE TABLE dbo.Fact_FacilityRequest_Partitioned
(
   FacilityReqKey BIGINT NOT NULL,
   DateKey INT NOT NULL,
   UnitKey INT NOT NULL,
   JenisPermintaan VARCHAR(200) NOT NULL,
   Prioritas VARCHAR(20) NOT NULL,
   Status VARCHAR(50) NOT NULL,

   CONSTRAINT FK_FacReqP_Date
      FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey),
   CONSTRAINT FK_FacReqP_Unit
      FOREIGN KEY (UnitKey) REFERENCES dbo.Dim_Unit(UnitKey)
)
ON PS_DateKey_Year(DateKey);
GO

ALTER TABLE dbo.Fact_FacilityRequest_Partitioned
ADD CONSTRAINT PK_Fact_FacilityRequest_Partitioned
PRIMARY KEY CLUSTERED (FacilityReqKey, DateKey)
ON PS_DateKey_Year(DateKey);
GO

CREATE TABLE dbo.Fact_ItemMovement_Partitioned
(
   MovementID BIGINT NOT NULL,
   ItemKey INT NOT NULL,
   DariRuangan INT NULL,
   KeRuangan INT NULL,
   DateKey INT NOT NULL,
   TglMutasi DATE NOT NULL,

   CONSTRAINT FK_MovementP_Item
```

```
    FOREIGN KEY (ItemKey) REFERENCES dbo.Dim_Item(ItemKey),
  CONSTRAINT FK_MovementP_Dari
    FOREIGN KEY (DariRuangan) REFERENCES dbo.Dim_Room(RoomKey),
  CONSTRAINT FK_MovementP_Ke
    FOREIGN KEY (KeRuangan) REFERENCES dbo.Dim_Room(RoomKey),
  CONSTRAINT FK_MovementP_Date
    FOREIGN KEY (DateKey) REFERENCES dbo.Dim_Date(DateKey)
)
ON PS_DateKey_Year(DateKey);
GO

ALTER TABLE dbo.Fact_ItemMovement_Partitioned
ADD CONSTRAINT PK_ItemMovement_Partitioned
PRIMARY KEY CLUSTERED (MovementID, DateKey)
ON PS_DateKey_Year(DateKey);
GO
```

Step 4: ETL Design

Tujuan: Merancang proses Extract, Transform, Load yang efisien

Aktivitas:

1. ETL Architecture Design
    - Source to Staging: Extract raw data

      Menarik data raw dari:
        - Sistem Peminjaman Ruangan
        - Sistem Maintenance / Perbaikan Barang
        - Sistem Permintaan Fasilitas
        - Master data Unit
        - Master data Ruangan
        - Master data Inventaris

      Tujuan:
        - Menyimpan raw data tanpa transformasi.
        - Menyimpan LoadDate untuk audit & recover.

    - Staging to Integration: Data cleansing dan transformation

      Proses:
        - Standarisasi kode unit/ruangan
        - Normalisasi nama barang/ruangan
        - Membersihkan null, duplicate
        - Mapping ke surrogate key untuk dimensi
        - Konversi format tanggal ke DateKey (YYYYMMDD)
        - Melakukan business rule (SLA, overtime, prioritization)

    - Integration to Data Warehouse: Load ke fact dan dimension tables
        - Load ke Dim Tables terlebih dahulu
        - Setelah dimensi stabil → load ke Fact Tables
        - Menggunakan partitioned fact tables (DateKey-based)
        - Menggunakan konsep SCD untuk dimensi tertentu

2. Create Staging Tables

```
-- Staging Schema
```

```
CREATE SCHEMA stg;
GO

CREATE TABLE stg.Dim_Date (
    DateKey INT,
    FullDate DATE,

    DayNumberOfWeek TINYINT,
    DayName VARCHAR(10),

    DayNumberOfMonth TINYINT,
    DayNumberOfYear SMALLINT,
    WeekNumberOfYear TINYINT,

    MonthName VARCHAR(20),
    MonthNumber TINYINT,

    Quarter TINYINT,
    QuarterName VARCHAR(10),
    Year SMALLINT,

    IsWeekend BIT,
    IsHoliday BIT,
    HolidayName VARCHAR(100),

    AcademicYear VARCHAR(9),
    Semester TINYINT,

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_Unit (
    NamaUnit VARCHAR(200),
    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_Gedung (
    GedungID INT,
    NamaGedung VARCHAR(200),
    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_RoomType (
    RoomTypeID INT,
    NamaTipeRuang VARCHAR(200),
    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_Room (
    RoomKey INT,
    RoomTypeID INT,
    RoomCode VARCHAR(50),
    NamaRuang VARCHAR(200),
```

```
    Kapasitas INT,
    GedungID INT,

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_ItemType (
    ItemTypeID INT,
    NamaJenisItem VARCHAR(200),
    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_KondisiItem (
    KondisiID INT,
    Kondisi VARCHAR(50),

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Dim_Item (
    ItemKey INT,
    ItemTypeID INT,
    NamaItem VARCHAR(200),
    KondisiID INT,
    CurrentRoomKey INT,

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Fact_RoomUsage (
    DateKey INT,
    RoomKey INT,
    UnitKey INT,

    DurationMinutes INT,
    SessionType VARCHAR(50),

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Fact_Repair (
    ItemKey INT,
    Status VARCHAR(50),
    DaysToComplete INT,
    DateKey INT,

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
```

```
CREATE TABLE stg.Fact_FacilityRequest (
    DateKey INT,
    UnitKey INT,

    JenisPermintaan VARCHAR(200),
    Prioritas VARCHAR(20),
    Status VARCHAR(50),

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE stg.Fact_ItemMovement (
    ItemKey INT,
    DariRuangan INT,
    KeRuangan INT,
    TglMutasi DATE,

    SourceSystem VARCHAR(50),
    LoadDate DATETIME DEFAULT GETDATE()
);
GO
```

3. ETL Mapping Document

Source	Source Column	Target	Target Column	Transformation
SARPRAS.dbo.Unit	NamaUnit	Dim_Unit	NamaUnit	UPPER(TRIM(NamaUnit))
SARPRAS.dbo.Gedung	NamaGedung	Dim_Gedung	NamaGedung	UPPER(TRIM(NamaGedung))
SARPRAS.dbo.RoomType	NamaTipeRuang	Dim_RoomType	NamaTipeRuang	UPPER(TRIM(NamaTipeRuang))
SARPRAS.dbo.Room	RoomTypeID	Dim_Room	RoomTypeID	Direct mapping
SARPRAS.dbo.Room	RoomCode	Dim_Room	RoomCode	Direct mapping
SARPRAS.dbo.Room	NamaRuang	Dim_Room	NamaRuang	UPPER(TRIM(NamaRuang))
SARPRAS.dbo.Room	Kapasitas	Dim_Room	Kapasitas	Direct mapping
SARPRAS.dbo.Room	GedungID	Dim_Room	GedungID	Direct mapping
SARPRAS.dbo.ItemType	NamaJenisItem	Dim_ItemType	NamaJenisItem	UPPER(TRIM(NamaJenisItem))
SARPRAS.dbo.KondisiItem	Kondisi	Dim_KondisiItem	Kondisi	UPPER(TRIM(Kondisi))

Source	Source Column	Target	Target Column	Transformation
SARPRAS.dbo.Item	ItemTypeID	Dim_Item	ItemTypeID	Direct mapping
SARPRAS.dbo.Item	NamaItem	Dim_Item	NamaItem	UPPER(TRIM(NamaItem))
SARPRAS.dbo.Item	KondisiID	Dim_Item	KondisiID	Direct mapping
SARPRAS.dbo.Item	CurrentRoomKey	Dim_Item	CurrentRoomKey	Direct mapping
SARPRAS.dbo.Date	DateKey	Dim_Date	DateKey	Direct mapping
SARPRAS.dbo.Date	FullDate	Dim_Date	FullDate	Direct mapping
SARPRAS.dbo.Date	DayNumberOfWeek	Dim_Date	DayNumberOfWeek	Direct mapping
SARPRAS.dbo.Date	DayName	Dim_Date	DayName	UPPER(TRIM(DayName))
SARPRAS.dbo.Date	DayNumberOfMonth	Dim_Date	DayNumberOfMonth	Direct mapping
SARPRAS.dbo.Date	DayNumberOfYear	Dim_Date	DayNumberOfYear	Direct mapping
SARPRAS.dbo.Date	WeekNumberOfYear	Dim_Date	WeekNumberOfYear	Direct mapping
SARPRAS.dbo.Date	MonthName	Dim_Date	MonthName	UPPER(TRIM(MonthName))
SARPRAS.dbo.Date	MonthNumber	Dim_Date	MonthNumber	Direct mapping
SARPRAS.dbo.Date	Quarter	Dim_Date	Quarter	Direct mapping
SARPRAS.dbo.Date	QuarterName	Dim_Date	QuarterName	UPPER(TRIM(QuarterName))
SARPRAS.dbo.Date	Year	Dim_Date	Year	Direct mapping
SARPRAS.dbo.Date	IsWeekend	Dim_Date	IsWeekend	Direct mapping
SARPRAS.dbo.Date	IsHoliday	Dim_Date	IsHoliday	Direct mapping
SARPRAS.dbo.Date	HolidayName	Dim_Date	HolidayName	UPPER(TRIM(HolidayName))
SARPRAS.dbo.Date	AcademicYear	Dim_Date	AcademicYear	Direct mapping
SARPRAS.dbo.Date	Semester	Dim_Date	Semester	Direct mapping
SARPRAS.	DateKey	Fact_RoomUsage	DateKey	Direct mapping

Source	Source Column	Target	Target Column	Transformation
dbo.Room Usage				
SARPRAS. dbo.Room Usage	RoomKey	Fact_RoomUsage	RoomKey	Direct mapping
SARPRAS. dbo.Room Usage	UnitKey	Fact_RoomUsage	UnitKey	Direct mapping
SARPRAS. dbo.Room Usage	DurationMinutes	Fact_RoomUsage	DurationMinutes	Direct mapping
SARPRAS. dbo.Room Usage	SessionType	Fact_RoomUsage	SessionType	UPPER(TRIM(SessionType))
SARPRAS. dbo.Repair	ItemKey	Fact_Repair	ItemKey	Direct mapping
SARPRAS. dbo.Repair	Status	Fact_Repair	Status	UPPER(TRIM(Status))
SARPRAS. dbo.Repair	DaysToComplete	Fact_Repair	DaysToComplete	Direct mapping
SARPRAS. dbo.Repair	DateKey	Fact_Repair	DateKey	Direct mapping
SARPRAS. dbo.Facility Request	DateKey	Fact_FacilityRequest	DateKey	Direct mapping
SARPRAS. dbo.Facility Request	UnitKey	Fact_FacilityRequest	UnitKey	Direct mapping
SARPRAS. dbo.Facility Request	JenisPermintaan	Fact_FacilityRequest	JenisPermintaan	UPPER(TRIM(JenisPermintaan))
SARPRAS. dbo.Facility Request	Prioritas	Fact_FacilityRequest	Prioritas	UPPER(TRIM(Prioritas))
SARPRAS. dbo.Facility Request	Status	Fact_FacilityRequest	Status	UPPER(TRIM(Status))
SARPRAS. dbo.ItemMovement	ItemKey	Fact_ItemMovement	ItemKey	Direct mapping
SARPRAS. dbo.ItemMovement	DariRuangan	Fact_ItemMovement	DariRuangan	Direct mapping
SARPRAS. dbo.ItemM	KeRuangan	Fact_ItemMovement	KeRuangan	Direct mapping

Source	Source Column	Target	Target Column	Transformation
ovement				
SARPRAS.dbo.ItemMovement	TglMutasi	Fact_ItemMovement	TglMutasi	UPPER(TRIM(quarter_name))
SARPRAS.dbo.calendar	year	Dim_Date	Year	Direct mapping
SARPRAS.dbo.calendar	is_weekend	Dim_Date	IsWeekend	Direct mapping
SARPRAS.dbo.calendar	is_holiday	Dim_Date	IsHoliday	Direct mapping
SARPRAS.dbo.calendar	holiday_name	Dim_Date	HolidayName	UPPER(TRIM(holiday_name))
SARPRAS.dbo.calendar	academic_year	Dim_Date	AcademicYear	Direct mapping
SARPRAS.dbo.calendar	semester	Dim_Date	Semester	Direct mapping

Step 5: ETL Implementation

Tujuan: Mengimplementasikan ETL menggunakan SSIS atau T-SQL scripts

Opsi 1: Menggunakan SSIS (Recommended)

1. Create SSIS Project
    - Buka SQL Server Data Tools (SSDT)
    - Create new Integration Services Project
    - Beri nama: ETL_[UnitName]_DW
2. Package 1: Load Dimensions
    - Data Flow Task: Extract dari source
    - Derived Column: Transformasi data
    - Lookup Transformation: SCD Type 2 handling
    - Slowly Changing Dimension: Insert/Update dimension
3. Package 2: Load Facts
    - Data Flow Task: Extract enrollment data
    - Lookup Transformations: Resolve dimension keys
    - Derived Column: Calculate measures
    - OLE DB Destination: Load to fact table
4. Master Package
    - Sequence Container: Truncate staging
    - Execute SQL Task: Disable indexes
    - Execute Package Task: Load dimensions
    - Execute Package Task: Load facts

- Execute SQL Task: Rebuild indexes
- Execute SQL Task: Update statistics

Opsi 2: Menggunakan T-SQL Stored Procedures

```
-- Buat schema ETL
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'etl')
BEGIN
  EXEC('CREATE SCHEMA etl');
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_Unit
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_Unit AS tgt
  USING (
    SELECT DISTINCT NamaUnit, SourceSystem
    FROM stg.Dim_Unit
  ) AS src
    ON tgt.NamaUnit = src.NamaUnit
      AND tgt.IsCurrent = 1
  WHEN MATCHED AND (
    tgt.NamaUnit <> src.NamaUnit
  )
    THEN UPDATE
      SET tgt.IsCurrent = 0,
        tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (NamaUnit, SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.NamaUnit, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_Gedung
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_Gedung AS tgt
  USING (
    SELECT GedungID, NamaGedung, SourceSystem
    FROM stg.Dim_Gedung
  ) AS src
    ON tgt.GedungID = src.GedungID AND tgt.IsCurrent = 1
  WHEN MATCHED AND (tgt.NamaGedung <> src.NamaGedung)
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (GedungID, NamaGedung, SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.GedungID, src.NamaGedung, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_RoomType
AS
BEGIN
```

```
  SET NOCOUNT ON;

  MERGE dw.Dim_RoomType AS tgt
  USING (
    SELECT RoomTypeID, NamaTipeRuang, SourceSystem
    FROM stg.Dim_RoomType
  ) AS src
    ON tgt.RoomTypeID = src.RoomTypeID AND tgt.IsCurrent = 1
  WHEN MATCHED AND (tgt.NamaTipeRuang <> src.NamaTipeRuang)
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (RoomTypeID, NamaTipeRuang, SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.RoomTypeID, src.NamaTipeRuang, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_ItemType
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_ItemType AS tgt
  USING (
    SELECT ItemTypeID, NamaJenisItem, SourceSystem
    FROM stg.Dim_ItemType
  ) AS src
    ON tgt.ItemTypeID = src.ItemTypeID AND tgt.IsCurrent = 1
  WHEN MATCHED AND (tgt.NamaJenisItem <> src.NamaJenisItem)
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (ItemTypeID, NamaJenisItem, SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.ItemTypeID, src.NamaJenisItem, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_KondisiItem
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_KondisiItem AS tgt
  USING (
    SELECT KondisiID, Kondisi, SourceSystem
    FROM stg.Dim_KondisiItem
  ) AS src
    ON tgt.KondisiID = src.KondisiID AND tgt.IsCurrent = 1
  WHEN MATCHED AND (tgt.Kondisi <> src.Kondisi)
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (KondisiID, Kondisi, SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.KondisiID, src.Kondisi, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_Room
AS
BEGIN
  SET NOCOUNT ON;
```

```
  MERGE dw.Dim_Room AS tgt
  USING (
    SELECT RoomKey, RoomTypeID, RoomCode, NamaRuang, Kapasitas, GedungID, SourceSystem
    FROM stg.Dim_Room
  ) AS src
    ON tgt.RoomKey = src.RoomKey AND tgt.IsCurrent = 1
  WHEN MATCHED AND (
    tgt.RoomTypeID <> src.RoomTypeID OR
    tgt.RoomCode <> src.RoomCode OR
    tgt.NamaRuang <> src.NamaRuang OR
    tgt.Kapasitas <> src.Kapasitas OR
    tgt.GedungID <> src.GedungID
  )
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (RoomKey, RoomTypeID, RoomCode, NamaRuang, Kapasitas, GedungID,
          SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.RoomKey, src.RoomTypeID, src.RoomCode, src.NamaRuang,
          src.Kapasitas, src.GedungID, src.SourceSystem,
          GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_Item
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_Item AS tgt
  USING (
    SELECT ItemKey, ItemTypeID, NamaItem, KondisiID, CurrentRoomKey, SourceSystem
    FROM stg.Dim_Item
  ) AS src
    ON tgt.ItemKey = src.ItemKey AND tgt.IsCurrent = 1
  WHEN MATCHED AND (
    tgt.ItemTypeID <> src.ItemTypeID OR
    tgt.NamaItem <> src.NamaItem OR
    tgt.KondisiID <> src.KondisiID OR
    tgt.CurrentRoomKey <> src.CurrentRoomKey
  )
    THEN UPDATE SET tgt.IsCurrent = 0, tgt.ExpiryDate = GETDATE()

  WHEN NOT MATCHED BY TARGET
    THEN INSERT (ItemKey, ItemTypeID, NamaItem, KondisiID, CurrentRoomKey,
          SourceSystem, EffectiveDate, ExpiryDate, IsCurrent)
      VALUES (src.ItemKey, src.ItemTypeID, src.NamaItem, src.KondisiID,
          src.CurrentRoomKey, src.SourceSystem, GETDATE(), NULL, 1);
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Dim_Date
AS
BEGIN
  SET NOCOUNT ON;

  MERGE dw.Dim_Date AS tgt
  USING stg.Dim_Date AS src
    ON tgt.DateKey = src.DateKey
  WHEN NOT MATCHED BY TARGET
```

```
        THEN INSERT (
            DateKey, FullDate, DayNumberOfWeek, DayName, DayNumberOfMonth,
            DayNumberOfYear, WeekNumberOfYear, MonthName, MonthNumber,
            Quarter, QuarterName, Year, IsWeekend, IsHoliday,
            HolidayName, AcademicYear, Semester, SourceSystem
        )
        VALUES (
            src.DateKey, src.FullDate, src.DayNumberOfWeek, src.DayName,
            src.DayNumberOfMonth, src.DayNumberOfYear, src.WeekNumberOfYear,
            src.MonthName, src.MonthNumber, src.Quarter, src.QuarterName,
            src.Year, src.IsWeekend, src.IsHoliday, src.HolidayName,
            src.AcademicYear, src.Semester, src.SourceSystem
        );
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Fact_RoomUsage
AS
BEGIN
    INSERT INTO dw.Fact_RoomUsage (
        DateKey, RoomKey, UnitKey, DurationMinutes, SessionType, SourceSystem
    )
    SELECT DateKey, RoomKey, UnitKey, DurationMinutes, SessionType, SourceSystem
    FROM stg.Fact_RoomUsage;
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Fact_Repair
AS
BEGIN
    INSERT INTO dw.Fact_Repair (
        ItemKey, Status, DaysToComplete, DateKey, SourceSystem
    )
    SELECT ItemKey, Status, DaysToComplete, DateKey, SourceSystem
    FROM stg.Fact_Repair;
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Fact_FacilityRequest
AS
BEGIN
    INSERT INTO dw.Fact_FacilityRequest (
        DateKey, UnitKey, JenisPermintaan, Prioritas, Status, SourceSystem
    )
    SELECT DateKey, UnitKey, JenisPermintaan, Prioritas, Status, SourceSystem
    FROM stg.Fact_FacilityRequest;
END;
GO

CREATE OR ALTER PROCEDURE etl.Load_Fact_ItemMovement
AS
BEGIN
    INSERT INTO dw.Fact_ItemMovement (
        ItemKey, DariRuangan, KeRuangan, TglMutasi, SourceSystem
    )
    SELECT ItemKey, DariRuangan, KeRuangan, TglMutasi, SourceSystem
    FROM stg.Fact_ItemMovement;
END;
GO

CREATE OR ALTER PROCEDURE etl.Master_ETL_Sarpras
```

```
AS
BEGIN
  PRINT 'START ETL...';

  EXEC etl.Load_Dim_Unit;
  EXEC etl.Load_Dim_Gedung;
  EXEC etl.Load_Dim_RoomType;
  EXEC etl.Load_Dim_ItemType;
  EXEC etl.Load_Dim_KondisiItem;
  EXEC etl.Load_Dim_Room;
  EXEC etl.Load_Dim_Item;
  EXEC etl.Load_Dim_Date;

  EXEC etl.Load_Fact_RoomUsage;
  EXEC etl.Load_Fact_Repair;
  EXEC etl.Load_Fact_FacilityRequest;
  EXEC etl.Load_Fact_ItemMovement;

  PRINT 'ETL COMPLETED.';
END;
GO
```

	LogID	ProcedureName	StartTime	EndTime	Status	Remarks
1	15	Load_Fact_ItemMovement	2025-11-24 15:09:03.883	2025-11-24 15:09:03.907	SUCCESS	NULL
2	14	Load_Fact_FacilityRequest	2025-11-24 15:09:03.867	2025-11-24 15:09:03.870	SUCCESS	NULL
3	13	Load_Fact_Repair	2025-11-24 15:09:03.853	2025-11-24 15:09:03.853	SUCCESS	NULL
4	12	Load_Fact_RoomUsage	2025-11-24 15:09:03.840	2025-11-24 15:09:03.840	SUCCESS	NULL
5	11	Load_Dim_Date	2025-11-24 15:09:03.787	2025-11-24 15:09:03.793	SUCCESS	NULL
6	10	Load_Dim_Item	2025-11-24 15:09:03.777	2025-11-24 15:09:03.780	SUCCESS	NULL
7	9	Load_Dim_Room	2025-11-24 15:09:03.763	2025-11-24 15:09:03.763	SUCCESS	NULL
8	8	Load_Dim_KondisiItem	2025-11-24 15:09:03.743	2025-11-24 15:09:03.743	SUCCESS	NULL
9	7	Load_Dim_ItemType	2025-11-24 15:09:03.733	2025-11-24 15:09:03.737	SUCCESS	NULL
10	6	Load_Dim_RoomType	2025-11-24 15:09:03.723	2025-11-24 15:09:03.723	SUCCESS	NULL
11	5	Load_Dim_Gedung	2025-11-24 15:09:03.697	2025-11-24 15:09:03.697	SUCCESS	NULL
12	4	Load_Dim_Unit	2025-11-24 15:09:03.677	2025-11-24 15:09:03.687	SUCCESS	NULL
13	3	Load_Dim_Unit	2025-11-24 15:07:15.267	2025-11-24 15:07:15.300	SUCCESS	NULL
14	2	Load_Dim_Unit	2025-11-24 15:06:35.823	2025-11-24 15:06:35.827	SUCCESS	NULL
15	1	Load_Dim_Unit	2025-11-24 14:54:43.783	NULL	STARTED	NULL

Step 6: Data Quality Assurance

Tujuan: Memastikan kualitas data yang dimuat ke data warehouse

Aktivitas:

```
-- Dim_Unit
SELECT 'Dim_Unit' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN NamaUnit IS NULL THEN 1 ELSE 0 END) AS Null_NamaUnit
FROM dw.Dim_Unit;

-- Dim_Gedung
SELECT 'Dim_Gedung' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN GedungID IS NULL THEN 1 ELSE 0 END) AS Null_GedungID,
    SUM(CASE WHEN NamaGedung IS NULL THEN 1 ELSE 0 END) AS Null_NamaGedung
FROM dw.Dim_Gedung;

-- Dim_RoomType
SELECT 'Dim_RoomType' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN RoomTypeID IS NULL THEN 1 ELSE 0 END) AS Null_RoomTypeID,
    SUM(CASE WHEN NamaTipeRuang IS NULL THEN 1 ELSE 0 END) AS Null_NamaTipeRuang
FROM dw.Dim_RoomType;
```

```
-- Dim_Room
SELECT 'Dim_Room' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN RoomKey IS NULL THEN 1 ELSE 0 END) AS Null_RoomKey,
    SUM(CASE WHEN RoomCode IS NULL THEN 1 ELSE 0 END) AS Null_RoomCode,
    SUM(CASE WHEN NamaRuang IS NULL THEN 1 ELSE 0 END) AS Null_NamaRuang,
    SUM(CASE WHEN GedungID IS NULL THEN 1 ELSE 0 END) AS Null_GedungID
FROM dw.Dim_Room;

-- Dim_ItemType
SELECT 'Dim_ItemType' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN ItemTypeID IS NULL THEN 1 ELSE 0 END) AS Null_ItemTypeID,
    SUM(CASE WHEN NamaJenisItem IS NULL THEN 1 ELSE 0 END) AS Null_NamaJenisItem
FROM dw.Dim_ItemType;

-- Dim_KondisiItem
SELECT 'Dim_KondisiItem' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN KondisiID IS NULL THEN 1 ELSE 0 END) AS Null_KondisiID,
    SUM(CASE WHEN Kondisi IS NULL THEN 1 ELSE 0 END) AS Null_Kondisi
FROM dw.Dim_KondisiItem;

-- Dim_Item
SELECT 'Dim_Item' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN ItemKey IS NULL THEN 1 ELSE 0 END) AS Null_ItemKey,
    SUM(CASE WHEN ItemTypeID IS NULL THEN 1 ELSE 0 END) AS Null_ItemTypeID,
    SUM(CASE WHEN KondisiID IS NULL THEN 1 ELSE 0 END) AS Null_KondisiID,
    SUM(CASE WHEN CurrentRoomKey IS NULL THEN 1 ELSE 0 END) AS Null_CurrentRoomKey
FROM dw.Dim_Item;

-- Dim_Date
SELECT 'Dim_Date' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN DateKey IS NULL THEN 1 ELSE 0 END) AS Null_DateKey,
    SUM(CASE WHEN FullDate IS NULL THEN 1 ELSE 0 END) AS Null_FullDate
FROM dw.Dim_Date;

-- Fact_RoomUsage
SELECT 'Fact_RoomUsage' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN DateKey IS NULL THEN 1 ELSE 0 END) AS Null_DateKey,
    SUM(CASE WHEN RoomKey IS NULL THEN 1 ELSE 0 END) AS Null_RoomKey,
    SUM(CASE WHEN UnitKey IS NULL THEN 1 ELSE 0 END) AS Null_UnitKey
FROM dw.Fact_RoomUsage;

-- Fact_Repair
SELECT 'Fact_Repair' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN ItemKey IS NULL THEN 1 ELSE 0 END) AS Null_ItemKey,
    SUM(CASE WHEN DateKey IS NULL THEN 1 ELSE 0 END) AS Null_DateKey
FROM dw.Fact_Repair;

-- Fact_FacilityRequest
SELECT 'Fact_FacilityRequest' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN DateKey IS NULL THEN 1 ELSE 0 END) AS Null_DateKey,
    SUM(CASE WHEN UnitKey IS NULL THEN 1 ELSE 0 END) AS Null_UnitKey
FROM dw.Fact_FacilityRequest;
```

```sql
-- Fact_ItemMovement
SELECT 'Fact_ItemMovement' AS TableName,
    COUNT(*) AS TotalRows,
    SUM(CASE WHEN ItemKey IS NULL THEN 1 ELSE 0 END) AS Null_ItemKey,
    SUM(CASE WHEN DariRuangan IS NULL THEN 1 ELSE 0 END) AS Null_DariRuangan,
    SUM(CASE WHEN KeRuangan IS NULL THEN 1 ELSE 0 END) AS Null_KeRuangan
FROM dw.Fact_ItemMovement;

-- Orphan RoomKey
SELECT COUNT(*) AS Orphan_RoomKey
FROM dw.Fact_RoomUsage f
LEFT JOIN dw.Dim_Room d ON f.RoomKey = d.RoomKey AND d.IsCurrent = 1
WHERE d.RoomKey IS NULL;

-- Orphan UnitKey
SELECT COUNT(*) AS Orphan_UnitKey
FROM dw.Fact_RoomUsage f
LEFT JOIN dw.Dim_Unit u ON f.UnitKey = u.UnitKey
WHERE u.UnitKey IS NULL;

-- Orphan DateKey
SELECT COUNT(*) AS Orphan_DateKey
FROM dw.Fact_RoomUsage f
LEFT JOIN dw.Dim_Date dt ON f.DateKey = dt.DateKey
WHERE dt.DateKey IS NULL;

SELECT COUNT(*) AS Orphan_ItemKey
FROM dw.Fact_Repair f
LEFT JOIN dw.Dim_Item i ON f.ItemKey = i.ItemKey AND i.IsCurrent = 1
WHERE i.ItemKey IS NULL;

SELECT COUNT(*) AS Orphan_DateKey
FROM dw.Fact_Repair f
LEFT JOIN dw.Dim_Date d ON f.DateKey = d.DateKey
WHERE d.DateKey IS NULL;

SELECT COUNT(*) AS Orphan_UnitKey
FROM dw.Fact_FacilityRequest f
LEFT JOIN dw.Dim_Unit u ON f.UnitKey = u.UnitKey
WHERE u.UnitKey IS NULL;

SELECT COUNT(*) AS Orphan_DateKey
FROM dw.Fact_FacilityRequest f
LEFT JOIN dw.Dim_Date d ON f.DateKey = d.DateKey
WHERE d.DateKey IS NULL;

SELECT COUNT(*) AS Orphan_ItemKey
FROM dw.Fact_ItemMovement f
LEFT JOIN dw.Dim_Item i ON f.ItemKey = i.ItemKey AND i.IsCurrent = 1
WHERE i.ItemKey IS NULL;

-- RoomUsage duration must be > 0
SELECT COUNT(*) AS Invalid_Duration
FROM dw.Fact_RoomUsage
WHERE DurationMinutes <= 0;

-- FacilityRequest Priority must be valid
SELECT COUNT(*) AS Invalid_Prioritas
FROM dw.Fact_FacilityRequest
WHERE Prioritas NOT IN ('Low','Medium','High');
```

```
-- Repair DaysToComplete must be >= 0
SELECT COUNT(*) AS Invalid_RepairDays
FROM dw.Fact_Repair
WHERE DaysToComplete < 0;

-- Duplicate RoomUsage (DateKey, RoomKey, UnitKey)
SELECT DateKey, RoomKey, UnitKey,
    COUNT(*) AS DuplicateCount
FROM dw.Fact_RoomUsage
GROUP BY DateKey, RoomKey, UnitKey
HAVING COUNT(*) > 1;

-- Duplicate ItemMovement for same item-date
SELECT ItemKey, TglMutasi,
    COUNT(*) AS DuplicateCount
FROM dw.Fact_ItemMovement
GROUP BY ItemKey, TglMutasi
HAVING COUNT(*) > 1;

-- RoomUsage
SELECT 'stg.Fact_RoomUsage' AS SourceTable, COUNT(*) AS RecordCount
FROM stg.Fact_RoomUsage
UNION ALL
SELECT 'dw.Fact_RoomUsage', COUNT(*)
FROM dw.Fact_RoomUsage;

-- Repair
SELECT 'stg.Fact_Repair', COUNT(*) FROM stg.Fact_Repair
UNION ALL
SELECT 'dw.Fact_Repair', COUNT(*) FROM dw.Fact_Repair;

-- FacilityRequest
SELECT 'stg.Fact_FacilityRequest', COUNT(*) FROM stg.Fact_FacilityRequest
UNION ALL
SELECT 'dw.Fact_FacilityRequest', COUNT(*) FROM dw.Fact_FacilityRequest;

-- ItemMovement
SELECT 'stg.Fact_ItemMovement', COUNT(*) FROM stg.Fact_ItemMovement
UNION ALL
SELECT 'dw.Fact_ItemMovement', COUNT(*) FROM dw.Fact_ItemMovement;
```

1. Create Data Quality Dashboard
    - Tabel audit untuk tracking data quality metrics
    - Stored procedure untuk generate quality report
    - Alert jika quality threshold tidak terpenuhi

	TableName	TotalRows	Null_NamaUnit
1	Dim_Unit	1000	0

	TableName	TotalRows	Null_GedungID	Null_NamaGedung
1	Dim_Gedung	1000	0	0

	TableName	TotalRows	Null_RoomTypeID	Null_NamaTipeRuang
1	Dim_RoomType	1000	0	0

	TableName	TotalRows	Null_RoomKey	Null_RoomCode	Null_NamaRuang	Null_GedungID
1	Dim_Room	1000	0	0	0	0

	TableName	TotalRows	Null_ItemTypeID	Null_NamaJenisItem
1	Dim_ItemType	1000	0	0

	TableName	TotalRows	Null_KondisiID	Null_Kondisi
1	Dim_KondisiItem	1000	0	0

	TableName	TotalRows	Null_ItemKey	Null_ItemTypeID	Null_KondisiID	Null_CurrentRoomKey
1	Dim_Item	1000	0	0	0	0

	TableName	TotalRows	Null_DateKey	Null_FullDate
1	Dim_Date	1000	0	0

	TableName	TotalRows	Null_DateKey	Null_RoomKey	Null_UnitKey
1	Fact_RoomUsage	1000	0	0	0

	TableName	TotalRows	Null_ItemKey	Null_DateKey
1	Fact_Repair	1000	0	0

	TableName	TotalRows	Null_DateKey	Null_UnitKey
1	Fact_FacilityRequest	1000	0	0

	TableName	TotalRows	Null_ItemKey	Null_DariRuangan	Null_KeRuangan
1	Fact_ItemMovement	1000	0	0	0

Step 7: Performance Testing

Tujuan: Menguji dan mengoptimalkan performa query

Aktivitas:

```
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

SELECT
   it.NamaJenisItem AS ItemCategory,
   COUNT(fr.RoomKey) AS TotalUsage,
   SUM(fr.DurationMinutes) AS TotalDuration
FROM dw.Fact_RoomUsage fr
INNER JOIN dw.Dim_Item i
   ON fr.RoomKey = i.CurrentRoomKey AND i.IsCurrent = 1
INNER JOIN dw.Dim_ItemType it
   ON i.ItemTypeID = it.ItemTypeID
INNER JOIN dw.Dim_Date d
   ON fr.DateKey = d.DateKey
WHERE d.Year = 2024
GROUP BY it.NamaJenisItem
ORDER BY TotalUsage DESC;

SELECT
   d.Year,
   d.MonthNumber,
   d.MonthName,
   COUNT(fr.RoomKey) AS TotalUsage,
   SUM(fr.DurationMinutes) AS TotalDuration
FROM dw.Fact_RoomUsage fr
INNER JOIN dw.Dim_Date d
   ON fr.DateKey = d.DateKey
GROUP BY
   d.Year, d.MonthNumber, d.MonthName
```

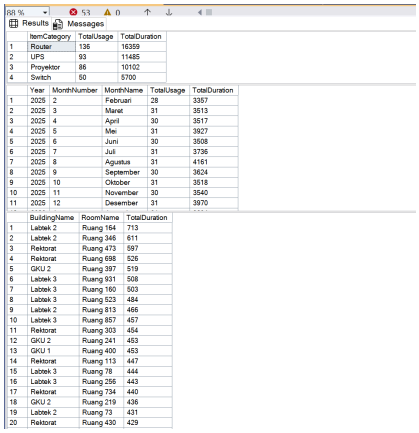
```
ORDER BY
    d.Year, d.MonthNumber;

SELECT
    g.NamaGedung AS BuildingName,
    r.NamaRuang AS RoomName,
    SUM(fr.DurationMinutes) AS TotalDuration
FROM dw.Fact_RoomUsage fr
INNER JOIN dw.Dim_Room r
    ON fr.RoomKey = r.RoomKey AND r.IsCurrent = 1
INNER JOIN dw.Dim_Gedung g
    ON r.GedungID = g.GedungID
GROUP BY
    g.NamaGedung, r.NamaRuang
ORDER BY
    TotalDuration DESC;

SELECT TOP 10
    i.NamaItem,
    it.NamaJenisItem AS ItemCategory,
    SUM(fr.DurationMinutes) AS TotalDuration
FROM dw.Fact_RoomUsage fr
INNER JOIN dw.Dim_Item i
    ON fr.RoomKey = i.CurrentRoomKey AND i.IsCurrent = 1
INNER JOIN dw.Dim_ItemType it
    ON i.ItemTypeID = it.ItemTypeID
GROUP BY
    i.NamaItem, it.NamaJenisItem
ORDER BY
    TotalDuration DESC;
```

1. Analyze Query Plans
    - Review execution plans
    - Identify missing indexes
    - Check for table scans
    - Optimize join orders

2. Performance Benchmarks



Results Messages

	ItemCategory	TotalUsage	TotalDuration
1	Router	136	16359
2	UPS	93	11485
3	Proyektor	86	10102
4	Switch	50	5700

	Year	MonthNumber	MonthName	TotalUsage	TotalDuration
1	2025	2	Februari	28	3357
2	2025	3	Maret	31	3513
3	2025	4	April	30	3617
4	2025	5	Mei	31	3927
5	2025	6	Juni	30	3608
6	2025	7	Juli	31	3736
7	2025	8	Agustus	31	4161
8	2025	9	September	30	3624
9	2025	10	Oktober	31	3518
10	2025	11	November	30	3540
11	2025	12	Desember	31	3970

	BuildingName	RoomName	TotalDuration
1	Labtek 2	Ruang 164	713
2	Labtek 2	Ruang 346	611
3	Rektorat	Ruang 473	597
4	Rektorat	Ruang 698	526
5	GKU 2	Ruang 397	519
6	Labtek 3	Ruang 931	508
7	Labtek 3	Ruang 160	503
8	Labtek 3	Ruang 523	484
9	Labtek 2	Ruang 813	466
10	Labtek 3	Ruang 857	457
11	Rektorat	Ruang 303	454
12	GKU 2	Ruang 241	453
13	GKU 1	Ruang 400	453
14	Rektorat	Ruang 113	447
15	Labtek 3	Ruang 78	444
16	Labtek 3	Ruang 256	443
17	Rektorat	Ruang 734	440
18	GKU 2	Ruang 219	436
19	Labtek 2	Ruang 73	431
20	Rektorat	Ruang 430	429

MISI 3: IMPLEMENTASI PRODUKSI

Step 1: Production Deployment
Tujuan: Men-deploy database dan ETL ke Azure VM production

Aktivitas:

1. Environment Setup
2. Database Deployment
3. Initial Data Load
4. Schedule ETL Jobs

```
USE msdb;
GO

-- 1. Create Job
EXEC sp_add_job
   @job_name     = N'ETL_Daily_Load_SARPRAS',
   @enabled      = 1,
   @description  = N'Daily ETL Load untuk Data Mart Sarpras';
GO

-- 2. Create Job Step (jalankan Master ETL)
EXEC sp_add_jobstep
   @job_name        = N'ETL_Daily_Load_SARPRAS',
   @step_name       = N'Execute Master ETL Sarpras',
   @subsystem       = N'TSQL',
   @command         = N'EXEC dbo.usp_Master_ETL_Sarpras;',
   -- Ganti jika nama SP berbeda
   @database_name   = N'DM_Sarpras_DW',
   @retry_attempts  = 3,
   @retry_interval  = 5;  -- menit
GO

-- 3. Create Daily Schedule (02:00 AM)
EXEC sp_add_schedule
   @schedule_name     = N'Sarpras Daily at 2 AM',
   @freq_type         = 4,        -- Daily
   @freq_interval     = 1,        -- Setiap 1 hari
   @active_start_time = 020000;   -- 02:00 AM (HHMMSS)
GO

-- 4. Attach Schedule to the Job
EXEC sp_attach_schedule
   @job_name      = N'ETL_Daily_Load_SARPRAS',
   @schedule_name = N'Sarpras Daily at 2 AM';
GO

-- 5. Register Job on Server
EXEC sp_add_jobserver
   @job_name = N'ETL_Daily_Load_SARPRAS';
GO
```

Step 2: Dashboard Development

Tujuan: Membangun interactive dashboard untuk end-users

Aktivitas:

1. Create Analytical Views

```
-- View: Room Utilization Summary
CREATE VIEW dbo.vw_Room_Utilization AS
SELECT
   dr.RoomKey,
   dr.NamaRuang,
   dr.RoomCode,
```

```
    dr.Kapasitas,
    dg.NamaGedung,
    drt.NamaTipeRuang,

    COUNT(fu.RoomUsageKey) AS TotalUsage,
    SUM(fu.DurationMinutes) AS TotalMinutesUsed,
    CAST(
        AVG(CAST(fu.DurationMinutes AS FLOAT))
        AS DECIMAL(10,2)
    ) AS AvgDuration,

    COUNT(DISTINCT fu.UnitKey) AS TotalUnitsUsing,

    -- Presentase penggunaan terhadap waktu 1 bulan (43.200 menit)
    CAST(
        SUM(fu.DurationMinutes) * 100.0 / NULLIF(43200, 0)
        AS DECIMAL(5,2)
    ) AS UtilizationRate

FROM dbo.Fact_RoomUsage fu
INNER JOIN dbo.Dim_Room dr ON fu.RoomKey = dr.RoomKey
INNER JOIN dbo.Dim_Gedung dg ON dr.GedungID = dg.GedungID
INNER JOIN dbo.Dim_RoomType drt ON dr.RoomTypeID = drt.RoomTypeID
GROUP BY
    dr.RoomKey,
    dr.NamaRuang,
    dr.RoomCode,
    dr.Kapasitas,
    dg.NamaGedung,
    drt.NamaTipeRuang;
GO

-- View: Facility Request Analytics
CREATE VIEW dbo.vw_Facility_Request_Analytics AS
SELECT
    du.NamaUnit,
    d.Year AS Tahun,
    d.MonthNumber AS Bulan,

    COUNT(fr.FacilityReqKey) AS TotalRequests,

    SUM(CASE WHEN fr.Prioritas = 'High' THEN 1 ELSE 0 END) AS HighPriority,
    SUM(CASE WHEN fr.Prioritas = 'Medium' THEN 1 ELSE 0 END) AS MediumPriority,
    SUM(CASE WHEN fr.Prioritas = 'Low' THEN 1 ELSE 0 END) AS LowPriority,

    SUM(CASE WHEN fr.Status = 'Completed' THEN 1 ELSE 0 END) AS Completed,
    SUM(CASE WHEN fr.Status = 'Pending' THEN 1 ELSE 0 END) AS Pending,
    SUM(CASE WHEN fr.Status = 'In Progress' THEN 1 ELSE 0 END) AS InProgress,

    CAST(
        SUM(CASE WHEN fr.Status = 'Completed' THEN 1 ELSE 0 END)
        * 100.0 / NULLIF(COUNT(*), 0)
        AS DECIMAL(5,2)
    ) AS CompletionRate

FROM dbo.Fact_FacilityRequest fr
INNER JOIN dbo.Dim_Unit du ON fr.UnitKey = du.UnitKey
INNER JOIN dbo.Dim_Date d ON fr.DateKey = d.DateKey
GROUP BY
    du.NamaUnit,
    d.Year,
```

```
    d.MonthNumber;
GO
```

2. Design Power BI Dashboards

Format Dashboard → beri warna berbeda untuk kategori prioritas, tipe ruangan, atau status

a. Dashboard 1: Executive Summary
   - KPI Cards
     - Total Items (Total Items) — hitung jumlah Dim_Item.ItemKey
     - Items Baik (Items - Baik) — count where Dim_KondisiItem.Kondisi = 'Baik'
     - Total Room Usage Minutes (Total Usage Minutes) — sum Fact_RoomUsage.DurationMinutes
     - Open Facility Requests (Open Requests) — count Fact_FacilityRequest where Status <> 'Closed'
   - Line Chart: Room usage trend (x = NameTypeRuangan / YearMonth, y = SUM(Fact_RoomUsage.DurationMinutes))
   - Bar Chart: Items per ItemType (x = Dim_ItemType.NamaJenisItem, y = count Dim_Item.ItemKey)
   - Map (jika ada lokasi) or Matrix: Room distribution by Gedung (Dim_Gedung.NamaGedung) — count rooms or items

b. Dashboard 2: Operational Performance
   - Stacked Bar: Repair Status by ItemType — axis ItemType; stacks = Status (Fact_Repair.Status), values = count(RepairKey)
   - Heat Map (matrix-style): Room utilization by Day — rows = DayName / RoomCode, columns = WeekNumber or Month, values = SUM(DurationMinutes)
   - Table: Top items with most repairs — ItemKey, NamaItem, Kondisi, #repairs

c. Dashboard 3: Maintenance & Logistics Analysis
   - Area Chart: Facility Request trend (per bulan) — x = Dim_Date.FullDate, y = count(FacilityReqKey)
   - Pie Chart: Requests by Prioritas — slice by Prioritas (High/Medium/Low)
   - Waterfall: Requests status flow (Open → In Progress → Closed) — build from counts per status
   - Forecast / Trend line: Projected requests next period — use built-in forecasting on request counts

d. Connect Power BI to SQL Server
   - Get Data → SQL Server
   - Server: [Azure VM IP/Hostname]
   - Database: DM_Sarpras_DW
   - Pilihan koneksi:
     - Import Mode – cocok untuk laporan statis
     - DirectQuery (disarankan untuk data penggunaan sarana yang real-time)
       - Load tabel seperti inventaris aset, log peminjaman, log perbaikan, serta view monitoring sarana
       - Dapat menggunakan DAX queries untuk agregasi kebutuhan sarana dan analisis performa penggunaan

3. Implement Interactivity
    - Slicers: Tahun Kegiatan, Jenis Sarana, Lokasi Penyimpanan, Status Kondisi
    - Drill-through: Dari ringkasan penggunaan sarana → detail aset per kategori atau per kegiatan
    - Cross-filtering: Antar visual saling memfilter, misalnya memilih kategori sarana akan otomatis memfilter grafik pemakaian, kondisi aset, dan biaya perawatan
    - Bookmarks: Untuk menampilkan berbagai tampilan seperti View Penggunaan, View Pemeliharaan, dan View Inventaris
    - Row-Level Security (opsional): Mengatur akses berdasarkan unit, misalnya staf Sarpras hanya melihat aset di bawah tanggung jawabnya

Step 3: Security Implementation
Tujuan: Mengimplementasikan access control dan audit trail
Aktivitas:

1. Create User Roles

```
-- Create Database Roles
CREATE ROLE db_executive;
CREATE ROLE db_analyst;
CREATE ROLE db_viewer;
CREATE ROLE db_etl_operator;
GO

-- Executive Full Access for SELECT and ETL Procedure
GRANT SELECT ON SCHEMA::dbo TO db_executive;
GRANT EXECUTE ON SCHEMA::dbo TO db_executive;
GO

-- Analyst: Can analyze and edit staging before loading
GRANT SELECT ON SCHEMA::dbo TO db_analyst;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO db_analyst;
GO

-- Viewer: Read-only access to all dimensional & fact tables
GRANT SELECT ON SCHEMA::dbo TO db_viewer;
GO

-- ETL Operator: Full access to staging + loading to DWH
GRANT EXECUTE ON SCHEMA::dbo TO db_etl_operator;
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::stg TO db_etl_operator;
GRANT INSERT, UPDATE ON SCHEMA::dbo TO db_etl_operator;
GO
```

2. Create Users and Assign Roles

```
-- Create SQL Logins
CREATE LOGIN executive_user WITH PASSWORD = 'Barcelona123';
CREATE LOGIN analyst_user WITH PASSWORD = 'Barcelona123';
CREATE LOGIN viewer_user WITH PASSWORD = 'Barcelona123';
CREATE LOGIN etl_service WITH PASSWORD = 'Barcelona123';
GO

-- Create Database Users
USE DM_SARPRAS_DW;
GO

CREATE USER executive_user FOR LOGIN executive_user;
CREATE USER analyst_user FOR LOGIN analyst_user;
CREATE USER viewer_user FOR LOGIN viewer_user;
CREATE USER etl_service FOR LOGIN etl_service;
```

```
GO

-- Assign Users to Roles
ALTER ROLE db_executive ADD MEMBER executive_user;
ALTER ROLE db_analyst ADD MEMBER analyst_user;
ALTER ROLE db_viewer ADD MEMBER viewer_user;
ALTER ROLE db_etl_operator ADD MEMBER etl_service;
GO
```

3. Implement Data Masking

```
-- Masking otomatis hanya jika kolom ditemukan

-- Nama Item (contoh jika NamaItem yang dipakai)
IF EXISTS (
   SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = 'Dim_Item' AND COLUMN_NAME = 'NamaItem'
)
BEGIN
   ALTER TABLE dbo.Dim_Item
   ALTER COLUMN NamaItem
   ADD MASKED WITH (FUNCTION = 'partial(1,"xxx",1)');
END

-- Deskripsi Item (alternative)
IF EXISTS (
   SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = 'Dim_Item' AND COLUMN_NAME = 'ItemDescription'
)
BEGIN
   ALTER TABLE dbo.Dim_Item
   ALTER COLUMN ItemDescription
   ADD MASKED WITH (FUNCTION = 'partial(1,"xxxxxxx",1)');
END

-- Serial Number / Code (default masking)
IF EXISTS (
   SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = 'Dim_Item' AND COLUMN_NAME = 'KodeItem'
)
BEGIN
   ALTER TABLE dbo.Dim_Item
   ALTER COLUMN KodeItem
   ADD MASKED WITH (FUNCTION = 'default()');
END
GO

-- Permission UNMASK
GRANT UNMASK TO db_executive;
GRANT UNMASK TO db_etl_operator;
GO
```

4. Implement Audit Trail
    - Create Audit Table

```
-- Create Audit Table
CREATE TABLE dbo.AuditLog (
   AuditID BIGINT IDENTITY(1,1) PRIMARY KEY,
   EventTime DATETIME2 DEFAULT SYSDATETIME(),
   UserName NVARCHAR(128) DEFAULT SUSER_SNAME(),
   EventType NVARCHAR(50),
   SchemaName NVARCHAR(128),
```

```
    ObjectName NVARCHAR(128),
    RowsAffected INT,
    SQLStatement NVARCHAR(MAX) NULL,
    HostName VARCHAR(128) NULL
);
GO
```

- Audit Trigger Example (Fact Repair)

```
CREATE TRIGGER trg_Audit_Fact_Repair
ON dbo.Fact_Repair
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @EventType NVARCHAR(50);

    IF EXISTS(SELECT * FROM inserted) AND EXISTS(SELECT * FROM deleted)
        SET @EventType = 'UPDATE';
    ELSE IF EXISTS(SELECT * FROM inserted)
        SET @EventType = 'INSERT';
    ELSE
        SET @EventType = 'DELETE';

    INSERT INTO dbo.AuditLog (EventType, SchemaName, ObjectName, RowsAffected)
    VALUES (@EventType, 'dbo', 'Fact_Repair', @@ROWCOUNT);
END;
GO
```

- Enable SQL Server Audit (Server-level)

```
USE master;
GO

IF EXISTS (SELECT * FROM sys.server_audits WHERE name = 'Sarpras_Audit')
    DROP SERVER AUDIT Sarpras_Audit;
GO

-- Buat audit target ke Application Log
CREATE SERVER AUDIT Sarpras_Audit
TO APPLICATION_LOG
WITH (ON_FAILURE = CONTINUE);
GO

ALTER SERVER AUDIT Sarpras_Audit
WITH (STATE = ON);
GO
```

- Create Database Audit Specification

```
CREATE DATABASE AUDIT SPECIFICATION Sarpras_DB_Audit
FOR SERVER AUDIT Sarpras_Audit
ADD (SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo BY public);
GO

ALTER DATABASE AUDIT SPECIFICATION Sarpras_DB_Audit
WITH (STATE = ON);
GO
```

Step 4: Backup and Recovery Strategy

Tujuan: Menyiapkan backup untuk disaster recovery

Aktivitas:

```
-- Full Backup
BACKUP DATABASE DM_Sarpras_DW
TO DISK = N'/var/opt/mssql/backup/DM_Sarpras_DW_Full.bak'
WITH
   COMPRESSION,
   INIT,
   NAME = 'Full Backup',
   STATS = 10;
GO

-- Differential Backup
BACKUP DATABASE DM_Sarpras_DW
TO DISK = N'/var/opt/mssql/backup/DM_Sarpras_DW_Diff.bak'
WITH DIFFERENTIAL, COMPRESSION, INIT, STATS = 10;
GO

-- Transaction Log Backup
SELECT name, recovery_model_desc
FROM sys.databases WHERE name='DM_Sarpras_DW';

ALTER DATABASE DM_Sarpras_DW SET RECOVERY FULL;
GO

BACKUP LOG DM_Sarpras_DW
TO DISK = N'/var/opt/mssql/backup/DM_Sarpras_DW_Log.trn'
WITH COMPRESSION, INIT, STATS = 10;
GO

-- Schedule Backup Jobs
-- Full Backup: Weekly (Sunday 2 AM)
-- Differential Backup: Daily (2 AM)
-- Transaction Log Backup: Every 6 hours

-- Backup to Azure Blob Storage (Optional)
CREATE CREDENTIAL [AzureStorageCredential]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
     SECRET = '<SAS_TOKEN>';
GO

BACKUP DATABASE DM_Sarpras_DW
TO URL = N'https://[storage_account].blob.core.windows.net/backups/DM_Sarpras_DW.bak'
WITH
   CREDENTIAL = 'AzureStorageCredential',
   COMPRESSION;
GO
```

Step 5: User Acceptance Testing

Tujuan: Validasi sistem dengan end-users

Aktivitas:

1. Create Test Cases
2. Conduct UAT Sessions
3. Performance Testing
4. Refinement

Step 6: Documentation

Tujuan: Menyiapkan dokumentasi lengkap untuk operasional

Dokumen yang Diperlukan:

1. System Architecture Document
2. Data Dictionary (Update dari Misi 1)
3. ETL Documentation
4. User Manual
5. Operations Manual
6. Security Documentation

Step 7: Final Presentation
Tujuan: Mempresentasikan hasil proyek kepada dosen dan stakeholders
Struktur Presentasi (20-25 menit):

1. Introduction
2. Requirements & Design
3. Implementation
4. Dashboard Demo
5. Technical Highlights
6. Lessons Learned & Future Work
7. Q&A