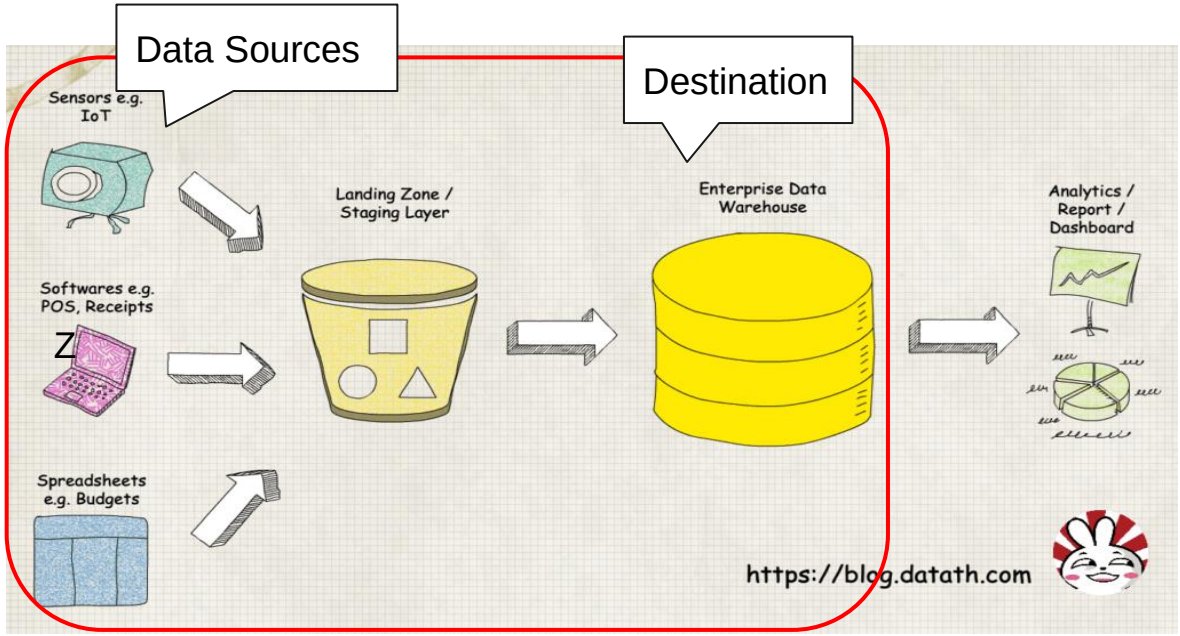
Chapter 1: Data Pipeline

มารู้จักกับ ETL และสร้าง Data Pipeline เพื่อเก็บข้อมูลจากฐานข้อมูล และ API

What is Data Pipeline

What is Data Pipeline?



ท่อในการลำเลียงข้อมูล จาก

แหล่งข้อมูล
(Data Source)

ไปยัง

จุดหมาย
(Destination)

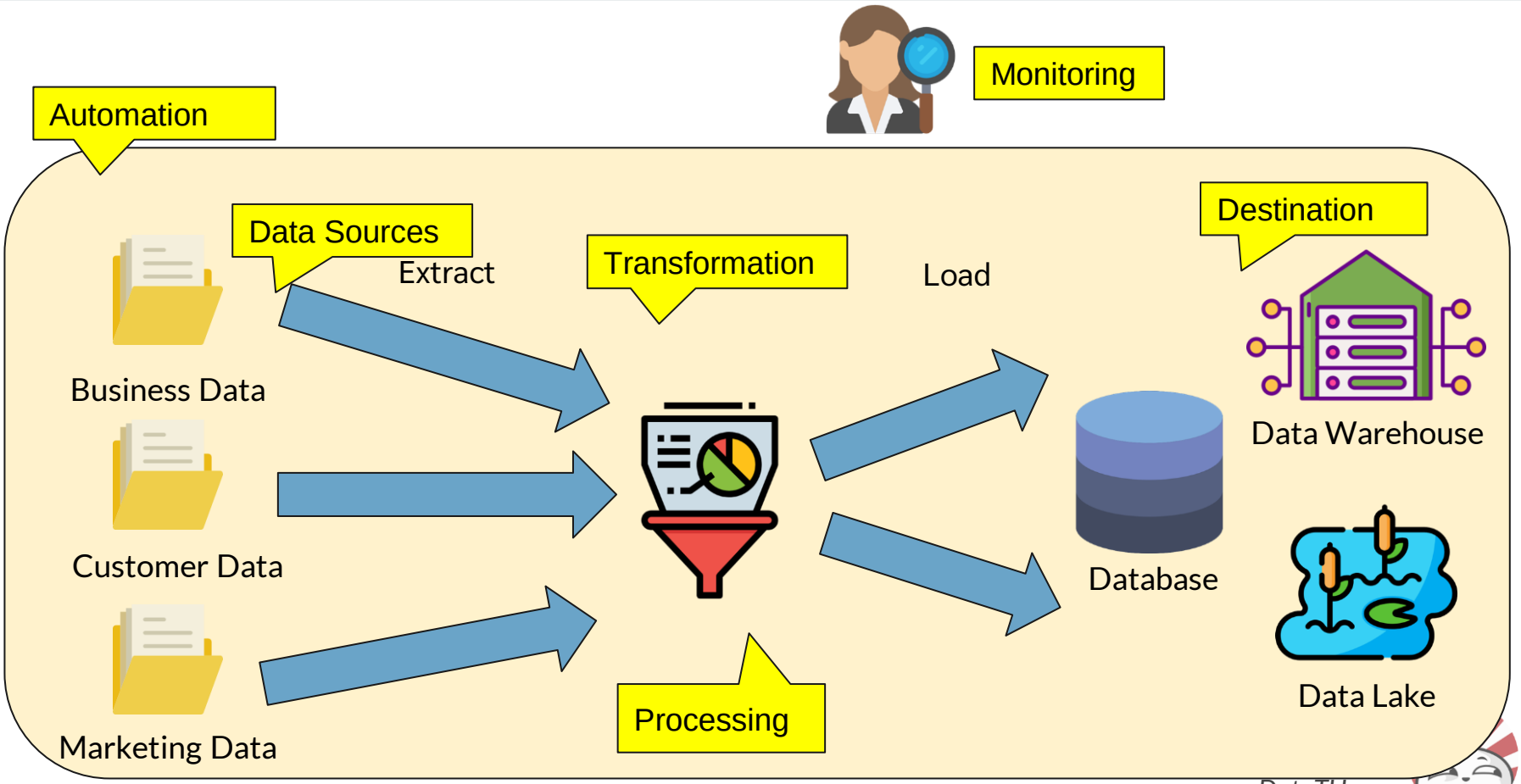
Why do we need Data Pipeline?

Locality
รวมข้อมูลเป็นหนึ่งเดียว

Decoupling
ไม่ต้องต่อท่อตรงจาก
Source ไป Destination

Data Pipeline Design

องค์ประกอบของ Data Pipeline



E = Extract

Business Data

Customer Data

Marketing Data

การดึงข้อมูลออกมาจากแหล่งข้อมูล (Data Source) ต่าง ๆ มีข้อที่ควรคำนึงถึงดังนี้:

- **ประเภทข้อมูล**
  เช่น CSV, JSON, API, Database, Data Warehouse / Mart ฯลฯ
- **หน้าตาข้อมูล**
  เช่น จำนวนคอลัมน์, ชื่อคอลัมน์, Format ของข้อมูลที่เก็บ (เช่น ชื่อ นามสกุล อาจเก็บรวมหรือแยกกัน)
- **ความถี่ในการอัพเดท**
  เช่น อัพเดทข้อมูลทุกชั่วโมง หรืออาทิตย์ละครั้ง

T = Transform

การเปลี่ยนแปลงข้อมูลสามารถทำได้หลากหลายรูปแบบ เช่น

- **ปรับให้รูปแบบเหมาะสมกับระบบปลายทาง**
  เช่น ต้นทางใช้วันที่แบบ DD/MM/YYYY ส่วนปลายทางใช้ YYYY-MM-DD

- **สรุปข้อมูล (Aggregation)**
  เช่น คำนวณค่าเฉลี่ย, ผลรวม

- **เพิ่มคุณค่าของข้อมูล (Enrichment)**
  เช่น รวมข้อมูลยอดขายของแต่ละวัน กับข้อมูลสภาพอากาศในวันนั้น ๆ

L = Load

Database

Data Warehouse

Data Lake

การนำข้อมูลเข้าไปในระบบปลายทาง

Database, Data Warehouse, Data Lake ก็สามารถเป็นระบบปลายทางได้ทั้งหมด

หากทำการ Transform มาก่อนหน้านี้ การ Load จะเป็นการส่งข้อมูลจากที่เก็บข้อมูลชั่วคราว (Staging Layer / Area) เข้าไปเก็บในระบบปลายทาง

ETL vs ELT

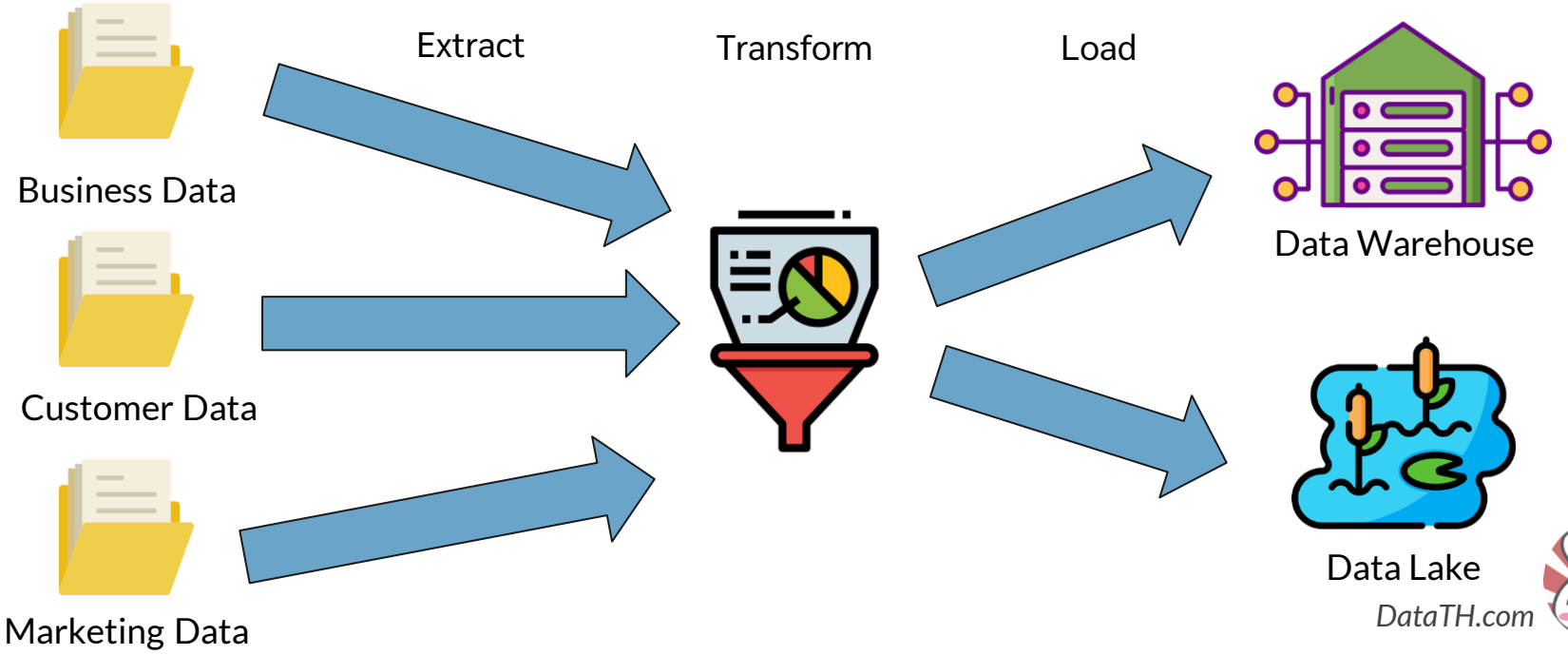
ETL - Extract - Transform - Load

- วิธีปกติในการย้ายข้อมูล เป็นที่นิยมในการย้ายข้อมูลไปที่ต่าง ๆ
- ระบบปลายทางไม่จำเป็นต้องประมวลผล (Transform) ข้อมูลเยอะ
- ต้องมี Staging Area แยก เพื่อประมวลผลข้อมูล
- Data Analyst ต้องรอ ETL เสร็จถึงจะได้ข้อมูล

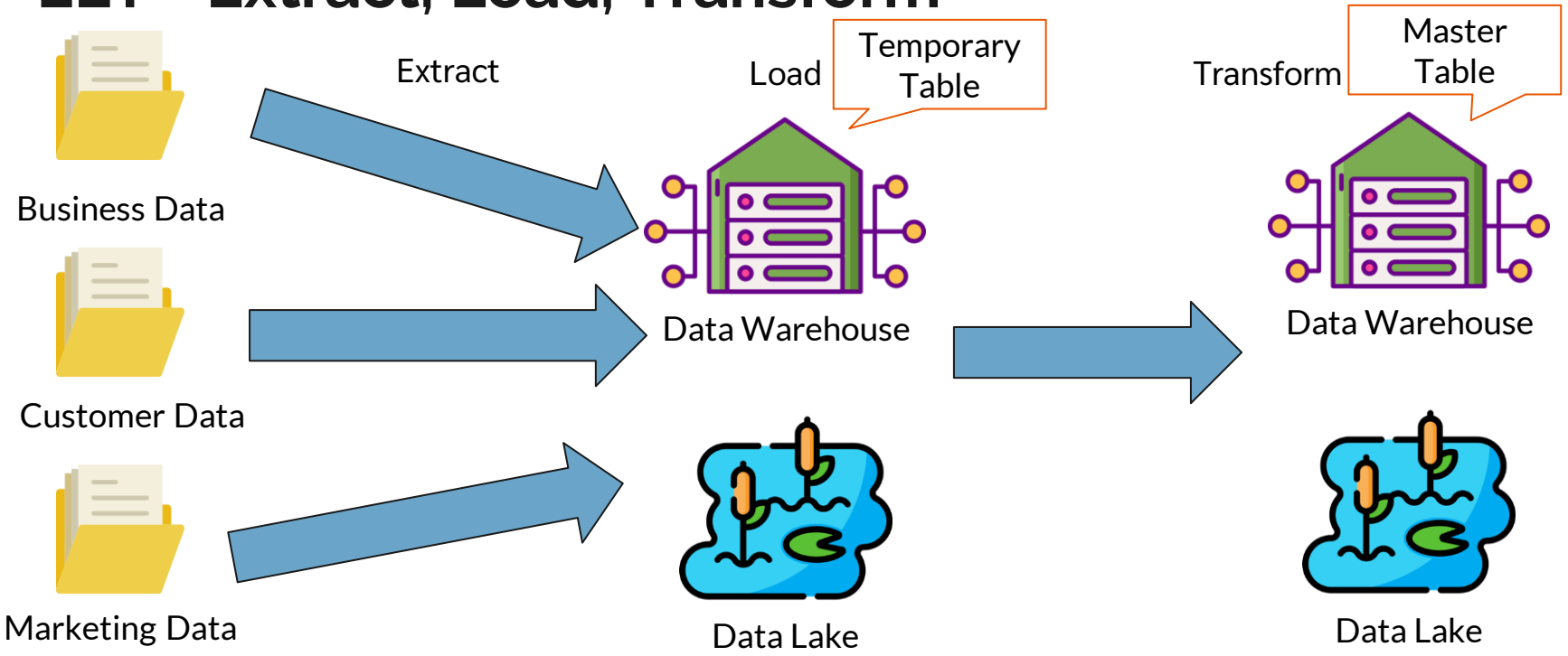
ELT - Extract - Load - Transform

- วิธีการย้ายข้อมูลสมัยใหม่ ระบบใหม่ ๆ จะสามารถทำได้ เช่น Redshift, Snowflake
- ระบบปลายทางจะต้องประมวลผลข้อมูลเยอะ
- ใช้ระบบปลายทางเป็น Staging Area
- Data Analyst เข้าถึงข้อมูลได้เร็วกว่า ไม่ต้องรอ ELT เสร็จ สามารถ Transform ข้อมูลดิบตอนดึงข้อมูลได้เลย

ETL - Extract, Transform, Load



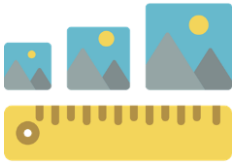
ELT - Extract, Load, Transform



Key Considerations / Trade-offs

Accuracy
ความถูกต้องของข้อมูล

Speed
ความเร็วในการย้ายข้อมูล



Scalability
ความสามารถในการรับข้อมูลปริมาณมาก

Security
ความปลอดภัยของข้อมูลระหว่างส่ง

Tip: การเพิ่มเรื่องหนึ่ง อาจจะไปลดอีกเรื่อง เช่น เพิ่ม Speed แล้ว Accuracy ลดลง เราต้องหาบาลานซ์ให้เหมาะสมกับความต้องการของโปรเจค

ประเภทของ Data Pipeline

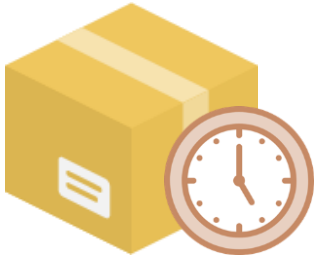
Initial Load / Historical load / Full load

ดึงข้อมูลทั้งหมดจากแหล่งข้อมูล

Incremental Load

ดึงข้อมูลใหม่ และข้อมูลที่เปลี่ยนแปลงจากครั้งล่าสุด

ประเภทของการประมวลผลข้อมูล (Processing)



Batch

ดึงตามช่วงเวลาที่กำหนด เช่น วันละครั้ง, ชั่วโมงละครั้ง

Stream

ข้อมูลจะถูกส่งเข้ามาทันที เราเลือกประมวลผลทันที หรือตามช่วงเวลาได้

Tip: เลือกตามความถี่ของแหล่งข้อมูลเป็นหลัก ข้อมูลบางแหล่งสามารถ Stream ได้ บางแหล่งไม่สามารถทำได้

เครื่องมือในการทำ ETL

No Code

Pre-built connectors

- Fivetran
- Stitch

Drag and drop

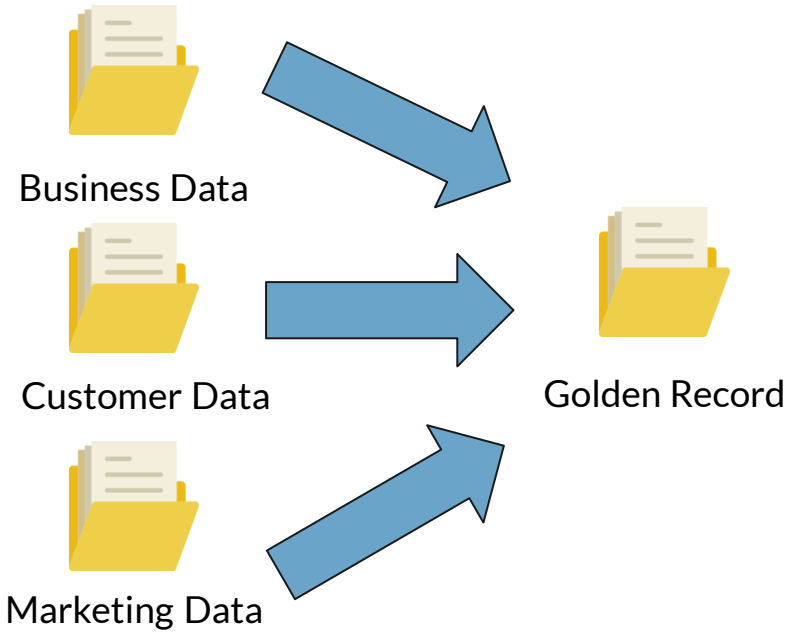
- Talend
- Informatica
- Azure Data Factory

Code

- Hadoop MapReduce
- Spark

Data Integration

Data integration คืออะไร



Data Integration เป็นการนำข้อมูลจากหลากหลายแหล่งข้อมูล และหลากหลายกฎทางธุรกิจ มารวมกันให้เป็นข้อมูลชุดเดียวที่มีประโยชน์กับองค์กร

ทำไม Data Integration ถึงมีประโยชน์

ช่วยให้เห็นภาพรวมของข้อมูลทั้งหมด

เช่น คุณ Gle เป็นลูกค้าของสินค้า A แต่ไม่ได้ใช้สินค้า B เราก็สามารถแนะนำสินค้า B ให้กับ Gle เพื่อเพิ่มรายได้ให้บริษัท

Informatica
Customer 360

https://www.informatica.com/products/master-data-management/customer-360.html

ข้อมูลมาจากไหนได้บ้าง

Database

Data Warehouse

Data Lake

Files

API - Free or $

Web Scraping

Types of data integration tasks

1) Schema integration

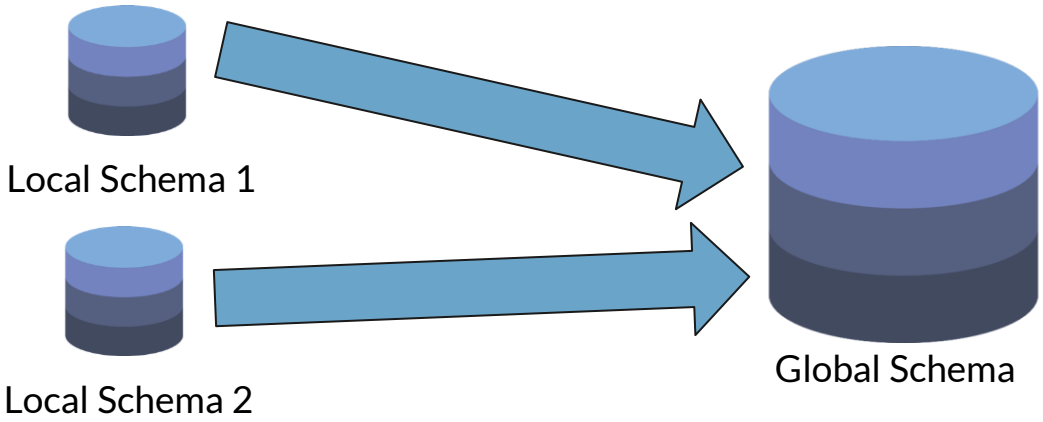
โครงสร้างข้อมูลแตกต่างกัน เช่น คอลัมน์ไม่เหมือนกัน, ใช้ชื่อเรียกต่างกัน, จัดกลุ่มข้อมูลไม่เหมือนกัน

2) Data integration

ข้อมูลเดียวกันแต่เก็บแตกต่างกัน เช่น ชื่อคน เก็บชื่อจริง กับชื่อเล่น (Jonathan กับ Jon)

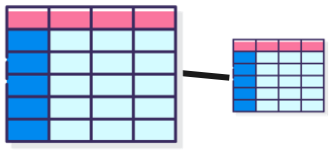
สิ่งที่ต้องทำใน Schema Integration

ต้องทำความเข้าใจโครงสร้างข้อมูลในแต่ละแหล่งข้อมูล (Local Schema) เพื่อนำมาสร้าง Global Schema



Tip: นัดคุยกับเจ้าของข้อมูลเพื่อสร้างความเข้าใจของแหล่งข้อมูลก่อน ถามคำถามสำคัญ เช่น มีคอลัมน์อะไรบ้าง อัพเดทบ่อยแค่ไหน หน้าตาข้อมูลเป็นยังไง

ปัญหาที่พบบ่อยในการทำ Schema Integration



Client ID
=
Customer ID

1.6 Kilometer
=
1 Miles

1. **Structure Conflict**
โครงสร้างข้อมูลไม่เหมือนกัน เช่น ระบบหนึ่งเก็บข้อมูลทุกอย่างใน 1 Table (Denormalised) อีกระบบเก็บข้อมูลแบบแยกเป็น 5 Table เล็ก ๆ (Normalised)

2. **Naming Conflict**
ข้อมูลเดียวกันแต่เรียกชื่อคอลัมน์ต่างกัน เช่น ระบบหนึ่งใช้ชื่อคอลัมน์ Client ID อีกระบบใช้ชื่อ Customer ID

3. **Entity Conflict**
ข้อมูลเดียวกันแต่เก็บคนละหน่วย เช่น ระยะทาง กิโลเมตร กับ ไมล์, ที่อยู่ระดับประเทศ กับ ระดับเมือง, ชื่อนำหน้า Mr. กับ Mister

Tip: แปลงข้อมูลให้เป็นหน่วยเดียวกันเสมอ และขอข้อมูลเพิ่มจากเจ้าของแหล่งข้อมูลก่อนทำการแก้ไข

ปัญหาที่พบบ่อยในการทำ Data Integration

Mister Jonathan
=
Mr. Jon

1. ข้อมูลซ้ำ ข้อมูลที่ค่าแตกต่างกันจากคนละแหล่งข้อมูล อาจจะหมายถึงข้อมูลเดียวกัน เช่น Mister Jonathan Holts กับ Mr. Jon อาจจะเป็นคนเดียวกัน ถ้ามีเบอร์โทรศัพท์เดียวกัน อีเมลเดียวกัน ที่อยู่เดียวกัน

2. ข้อมูลไม่ตรงกัน เกิดขึ้นได้บ่อยเมื่อรวมข้อมูลจากหลายระบบที่เวลาอัพเดทต่างกัน เช่น ที่อยู่จากข้อมูลที่อัพเดทปีละครั้ง กับที่อยู่จากข้อมูลที่อัพเดทเดือนละครั้ง จะไม่เหมือนกัน

Tip: จับคู่ข้อมูลจากหลายแหล่งข้อมูลโดยใช้คอลัมน์ที่มีค่าร่วมกัน (Primary Key) เช่น customer ID

Workshop 1:
Data Collection with Python

Workshop 1 - Data Collection with Python

เก็บข้อมูลจาก Database และ REST API ด้วย Python

Input:

- อ่านข้อมูลจาก MySQL
- อ่านข้อมูลจาก REST API ด้วย Package Requests

Output:

- Dataset ข้อมูลที่รวมแล้ว (CSV)