**BÁO CÁO**

**NỘI DUNG SÁCH**

# High-level Overview of an Analytics Setup

## Introduction

- Mục tiêu của sách:

+ Có 1 high-level framework, hiểu được setup về phân tích hiện đại thích hợp và cách các thành phần tương tác với nhau

+ Đi tìm hiểu chi tiết về từng thành phần, hiểu được vai trò của từng phần trong pipeline tổng cho dịch vụ data, bao gồm: tổng hợp (consolidating), chuyển đổi (transform), mô hình hóa (modeling) và sử dụng data

+ Hướng dẫn cách bắt đầu nhanh chóng một về setup phân tích và có thể thay đổi được theo thời gian

## A simple setup for people just starting out

Tất cả mọi hệ thống phân tích dữ liệu đều thực hiện 3 công việc cơ bản sau:

1, Bạn phải load data lên kho chứa trung tâm của data a.k.a data warehouse

2, Bạn sẽ phải chuyển đổi hoặc mô hình hóa dữ liệu của bạn để có thể sẵn sàng cho việc sử dụng chúng ở trong data warehouse

3, Bạn sẽ phải chuyển data đó đến người quyết định kinh doanh (business decision-makers)

## A modern analytics stack - một analytics stack hiện đại

Từ 3 công việc đã nói ở mục trên ta sẽ đi sâu chi tiết hơn:

Bước 1: Bạn sẽ phải thu thập, tập hợp và lưu trữ dữ liệu ở một kho dữ liệu trung tâm (central data warehouse)

2, Bạn sẽ phải tiến hành xử lý dữ liệu, bao gồm chuyển đổi, dọn dẹp, tổng hợp và mô hình hóa dữ liệu mà đã được đã đưa lên kho dữ liệu trung tâm

3, Bạn phải trình bày dữ liệu: visualize, extract, push dữ liệu lên các dịch vụ khác nhau hoặc cho người dùng cần đến chúng

## Mong muốn về một analytics stack tốt

- Thích *ELT* hơn ETL

- Ưu tiên sử dụng *một kho dữ liệu đám mây* (*cloud data warehouse*) hơn là một kho dữ liệu cố định tại chỗ (on-premise data warehouse)

- Ưu tiên *MPP analytics database* hơn là Hadoop system

- *Mô hình hóa dữ liệu* là rất cần thiết cho một anlytics setup và **không nên bỏ qua**

- *SQL based analytics* sẽ dành lợi thế so với non-SQL based analytics

- *Analytics workflow/operations* quan trọng hơn việc tập trung vào visualizations

# Centralizing Data

## Consolidating data from source systems

### Different types of source data

- Trong một tổ chức dữ liệu thường đến từ 3 nguồn chính sau:

**1, Dữ liệu trực tiếp từ ứng dụng chính (application database)**

=> Thường là các công ty mà các kinh doanh, dịch vụ của họ chạy ở trên website hay app.

=> Dữ liệu ở dạng này thường được lưu trữ trong một SQL (hoặc NoSQL) database được kết nối trực tiếp đến ứng dụng. Để truy cập thì thường sẽ cần kết nối chúng đến với một Database Query Tool.

**2, Dữ liệu từ Customer Relationship Management (CRM), Content Management System (CMS), Enterprise Resource Planning (ERP), or marketing systems**

=> Đây là những ứng dụng bên thứ ba được sử dụng trong việc điều hành của công ty

=> Những công cụ được quản lý bởi bên thứ ba nên thường chúng ta sẽ không truy cập trực tiếp vào dữ liệu lưu trữ trong đó. Bạn cần lấy dữ liệu qua API do công ty cung cấp và lưu trữ dữ liệu đó trong kho dữ liệu của riêng bạn

**3, Dữ liệu thủ công được tạo ra từ nhân viên và các hệ thống khác**

=> Vì những dữ liệu được tạo ra bởi con người không bằng những cấu trúc và quy định được thực thi nên thường gây ra lỗi

### The central datastore for analytics

- Những hệ thống trên là nơi lưu trữ và điều đình dữ liệu, không phải là nơi bạn chạy các các chương trình phân tích => Cần phải có một data warehouse

- Data warehouse là một central database lữu trữ và xử lý một lượng lớn dữ liệu cho mục đích phân tích. Một data warehouse có thể coi như là một database đã được tối ưu cho quy trình phân tích

- Data loading process là công việc trích xuất dữ liệu từ nhiều nguồn và tải chúng lên data warehouse => gọi là Extract & Load

### The Extract & Load process

- Extract & Load (EL) là một chương trình được viết để trích xuất dữ liệu thô từ nguồn dữ liệu và những dữ liệu sẽ được sao chép đến một địa chỉ đích.

- Trong công việc thực tế, việc viết đoạn code sẽ phức tạp hơn do có nhiều sự xem xét, đắn đo hơn, đồng thời chi phí để duy trì scripts sẽ đắt hơn giá trị thật mà nó mang lại

=> giải pháp ở dây là cân nhắc sử dụng một công cụ load data sẵn có

### Using a Data Loading Tool

- Có rất nhiều công cụ load data miễn phí và trả phí trên thị trường, thường sẽ hoạt động dưới dạng plug-and-play và có rất nhiều chức năng cần thiết cho người dùng. Những công cụ thường được biết đến là ETL tools, tuy nhiên người dùng cần phải sử dụng thêm 1 công cụ để chuyển đổi dữ liệu nữa để có thể hoàn thành được quá trình này.

### Common Concepts

*How do I load data incrementally?*

=> Đây là một khái niệm đáng để đề cập, khi mà hệ thống nguồn của bản phát triển về số lượng, concept này sẽ trở nên quan trọng cho việc đảm bảo hệ thống của bạn có thể chạy mượt mà. Hầu hết các công cụ tải dữ liệu trong nền công nghệ cơ bản đều nên hỗ trợ quy trình tải gia tăng (incremental load)

*How much performance gain does incremental load get you?*

=> Rất nhiều

*How often should I perform data loading?*

=> Hầu hết các doanh nghệp đều cần việc refresh cho dữ liệu hàng ngày

=> Trừ khi use case yêu cầu ngay lập tức thì không cần thiết phải cần những dữ liệu real-time trong công việc phân tích kinh doanh

## Understanding the Data Warehouse

### What is a data warehouse?

- Một data warehouse là một dạng cơ sở dữ liệu phân tích lưu trữ và xử lý dữ liệu cho mục đích phân tích. Một data warehouse sẽ đảm nhận 2 chức năng chính của việc phân tích: lưu trữ dữ liệu phân tích và xử lý chúng

- Chúng ta cần một data warehouse vì 2 mục đích sau:

1, Nếu dữ liệu nằm ở nhiều nguồn thì không thể tập hợp chúng một cách dễ dàng

2, Hệ thống nguồn không được thiết kế để có thể chạy những công việc phân tích có khối lượng lớn, nó có thể gây nguy hiểm cho hoạt động nghiệp vụ của bạn khi việc tải dữ liệu lên luôn có xu hướng tăng

- Data warehouse là trung tâm của mọi bước trong tiến trình luồng phân tích (analytics pipeline process) và nó phục vụ cho 3 mục đích chính:

* Storage => data warehouse sẽ nhận và lưu trữ data từ nhiều nguồn khác nhau
* Process => data warehouse sẽ đảm nhận quá trình xử lý chuyên sâu phát sinh từ bước chuyển đổi
* Access => các báo cáo sẽ được thu thập ở trong data warehouse trước sau đó phác hoạ (visualized) và chuyển đến end-users

### What is the right time to get a data warehouse?

=> Phụ thuộc vào những gì mà chúng ta có và cần

=> Để biết rằng data warehouse có phải là thứ chúng ta cần phải không thì có thể xem xét các luận điểm sau:

1, Có cần phải phân tích data từ nhiều nguồn?

2, Có cần phải tách rời data phân tích từ data kinh doanh/giao dịch?

3, Nguồn dữ liệu gốc có cho phép việc thực hiện truy vấn một cách hiệu quả?

4, Có mong muốn tăng hiệu quả cho công việc truy vấn phân tích?

### Which Data Warehouse Should I Pick?

### What makes a data warehouse different from normal SQL database?

=> Có thể dùng những cơ sở dữ liệu quan hệ để thay thế data warehouse hay không, cần hiểu một số khái niệm sau

***Transactional Workloads and Analytical Workloads***

- Khối công việc thuộc về giao dịch (transactional workloads) là khối công việc truy vấn phục vụ các ứng dụng kinh doanh thông thường

=> Các thuộc tính của transactional workloads:

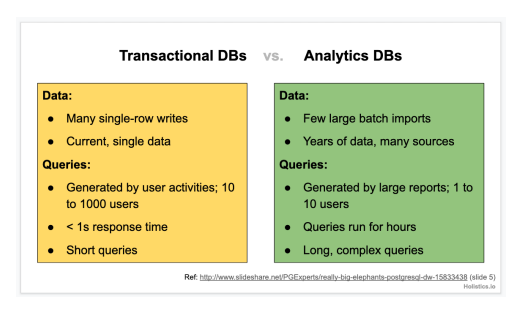
* Mỗi truy vấn thường chỉ có một bản ghi độc lập hoặc lượng nhỏ
* Transactional workloads thường gồm các truy vấn mất rất ít thời gian để chạy (< 1s)
* Nhiều truy vấn đồng thời bị giới hạn bởi số lượng người dùng truy cập cùng lúc vào ứng dụng
* Thường sẽ hứng thú với các record tổng

- Khối công việc phân tích (analytical workloads), hướng về mục đích phân tích. Khi một báo dữ liệu được chạy, một truy vấn sẽ được gửi tới DB để tính toán kết quả và hiển thị cho end-users

=> Các thuộc tính của analytical workloads:

* Mỗi truy vấn thường scan một lượng lớn các hàng trong bảng
* Mỗi truy vấn cần thời gian để có thể hoàn thành chạy (mins to hours)
* Không phải các truy vấn đồng thời đều diễn ra, bị giới hạn bởi lượng báo cáo hay internal staff sử dụng hệ thống phân tích
* Thường hứng thú chỉ với các dữ liệu trong một vài cột

=> Bảng so sánh về transactional DBs và analytics DBs:



***The Backend for Analytics Database is Different***

## ETL vs. ELT – What’s the big deal?

### The ETL process

- ETL (Extract, Transform and Load) gồm 3 bước. Trong quy trình này, một công cụ ETL trích xuất dữ liệu từ nhiều hệ thống nuồn dữ liệu khác nhau, chuyển đổi dữ liệu bằng các công cụ tính toán, nối lại và cuối cùng tải dữ liệu lên data warehouse.

- Chi tiết về cách tiếp cận này:

1, Trích xuất dữ liệu từ các nguồn (viết truy vấn, gửi yêu cầu trích xuất tới úng dụng thông qua API)

2, Các dữ liệu đã được trích xuất sẽ được biến đổi ở trong bộ nhớ của công cụ ETL

3, Các dữ liệu đã được biến đổi sẽ được tải lên data warehouse cuối cùng

=> Key things: dữ liệu thô được biến đổi bên ngoài data warehouse

### The ELT process

### - ELT là con đường có góc nhìn khác về vấn đề. Thay vì biến đổi dữ liệu trước khi tải lên database, ELT tiến hành bước chuyển đổi ở trong data warehouse. Dữ liệu sẽ được tải lên data warehouse trước rồi mới được biến đổi tại đây.

- Chi tiết về cách tiếp cận này:

1, Trích xuất dữ liệu từ các nguồn

2, Thay vì biến đổi trong bộ nhớ, chúng ta sử dụng công cụ phân luồn, tải các dữ liêu thô, được trích xuất lên thẳng data warehouse đích

3, Tiến hành thực hiện các biến đổi cần thiết ở trong data warehouse

=> Key things: dữ liệu thô được biến đổi bên trong data warehouse mà không cần có một dàn máy chủ, bây giờ data warehouse chứa cả dữ liệu thô và dữ liệu đã được biến đổi

### The shift from ETL to ELT

- Việc xây dựng một data warehouse rất tốn kém

- Hướng tiếp cận ETL có một số khuyết điểm khi nhìn nhận từ một góc nhìn đương thời:

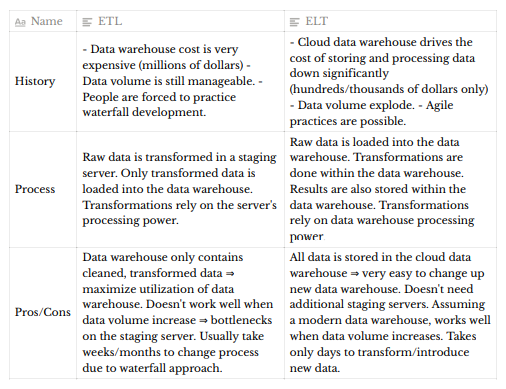
* Tất cả các biến đổi đều yêu cầu những đóng góp về IT hoặc từ data engineering để có thể update code cho việc biến đổi
* Với việc data size ngày càng tăng, hướng tiếp cân ETL sẽ gặp nhiều vấn đề, đặc biệt có thể tạo ra hiện tượng nút cổ chai khi staging server phải đảm nhận toàn bộ việc tải và biến đổi dữ liệu

=> Gần đây, cloud data warehouse xuất hiện ngày càng nhiều, cách mạng big data, sự tăng lên trong việc áp dụng mô hình phát triển agile => giá thành lưu trữ và xử lý dữ liệu trở nên rất rẻ => ý tưởng bây giờ là chuyển toàn bộ dữ liệu lên một địa chỉ trung tâm trước khi áp dụng bất kì biến đổi nào

- Hướng tiếp cận ELT có những ưu điểm sau:

* Cắt giảm hiện tượng thắt nút cổ chai ở máy chủ/ETL pipelining tool
* Không yêu cầu chi tiết về việc dữ liệu sẽ biến đổi thành cái gì
* Không yêu cầu data engineers phải luôn hỗ trợ bất kì yêu cầu từ team phân tích

=> Đây là bảng tóm tắt những điểm khác nhau giữa ETL và ELT:



### What about Data Lakes?

- Data lake là một điều thú vị cho khu vực dàn máy trung tâm cho dữ liệu thô

- Data lake không phải là một ý tưởng mới

## Transforming Data in the ELT paradigm

### What is data transformation?

- Biến đổi dữ liệu là quá trình mà thay đổi cấu trúc của dữ liệu hiện tại sang một cấu trúc khác

- Các use case phổ biến của biến đổi dữ liệu bao gồm:

* Data cleaning
* Aggregate data
* Pre-computation

### Why do I need to transform data?

- Để việc phân tích dữ liệu hiệu quả, chuyển dổi là bước cần để làm cho việc đó trở nên dễ hơn a.k.a data modeling

- Áp dụng việc biến đổi dữ liệu tốt sẽ đem lại các lợi ích sau:

* Reusability
* Your reports and analyses will be more consistent
* Improve overall runtime performance
* Cost effectiveness

### Implementing data transformation in the ELT paradigm

Việc dữ liệu được tải lên trước khi biến đổi sẽ cho phép chúng ta làm hai việc:

1, Cho phép chúng ta viết biến đổi chỉ sử dụng SQL

2, Cho phép các nhà phân tích viết các biến đổi => giảm sự phụ thuộc vào DE và cho phép xây dựng và duy trì pipelines ở external tools

### Using data transform tools

- Trong thực tế, sử dụng các công cụ biến đổi tốt sẽ giúp xử lý các bản mẫu SQL và để chúng ta tập trung vào core transformation logic.

### The transformation workflow (or the DAG)

- Khi cần phần xử lý các biến đổi có sự phụ thuộc lẫn nhau, chúng ta sẽ gặp phải vấn đề quản lý sự phụ thuộc => concept này có tên là DAG a.k.a directed acyclic graph workflow.

### A note on traditional Data Transform using ETL

- Các biến đổi dữ liệu được sử dụng theo cách truyền thống sẽ diễn ra và được trình bày trong một ngôn ngữ lập trình, trước khi dữ liệu được tải lên data warehouse

- Vấn đề chính của cách tiếp cận này chính là việc phần lớn việc tải lên chỉ diễn ra tại một máy tính chạy đoạn script đó

### Advanced Topic: Incremental transform to optimize for performance

- Giống với Incremental Load, chỉ có những gì khác biệt trong dữ liệu của điểm đầu và điểm cuối mới được tải lên.

- Tuy nhiên, phải hiểu rõ rằng không phải lúc nào cũng có thể chạy việc biến đổi gia tăng khi:

* Dữ liệu cũ vẫn có thể thay đổi
* Việc quá trình chuyển đổi/tải cũng cần phải mang tính chất gia tăng => Không phải lúc nào cũng chính xác

# Data Modeling for Analytics

## Data Modeling Layer & Concepts

### A contemporary look at data modeling

### What is data modeling and why is it needed?

- DA hiểu đơn giản sẽ là 1 “data translator” trong đó:

* Nhận những yêu cầu từ phía staff
* Tiến hành tìm những dữ liệu thích hợp có trong data warehouse
* Chuyển những câu hỏi, yêu cầu về kinh doanh sang những logic về dữ liệu tương ứng và diễn giải những logic này trong các form truy vấn (dùng SQL)
* Tiến hành chạy các truy vấn, lấy kết quả và chuyển sang định dạng mà staff yêu cầu

=> DA phải hiểu rõ về ánh xạ giữa business logic và data logic

=> Tuy nhiên quá trình này chỉ thực hiện tốt ở 1 số công ty bởi nếu nhiều người cùng yêu cầu sẽ dẫn đến những khó khăn sau:

* DA sẽ trở thành một ‘nút thắt cổ chai’ => Tất cả mọi yêu cầu từ nhỏ đến to đều phải qua DA
* Khi DA nghỉ việc hay quên việc cài đặt trung tâm của business logic
* Thời gian chờ đợi lâu
* DA sẽ mất thời gian để hoàn thành yêu cầu của mọi người, thay vì tập trung vào công việc chính của họ

=> Ý tưởng sẽ là đem những tri thức của DA sang một hệ thống mà tất cả mọi người

### The Data Modeling Layer

- Những gì chúng ta muốn là tiến hành một vài chuỗi biến đổi để đem những tri thức ánh xạ của DA thanh những thứ gắn liền trong data warehouse => Toàn bộ những biến đổi này sẽ được chạy trong data warehouse a.k.a quy trình ELT

- Hướng tiếp cận hiện đại để làm công việc data modeling là sắp xếp việc biến đổi đó ở trong data warehouse, thông qua một công cụ nằm phía trên cùng của data warehouse

=> Những công cụ như này có những đặc điểm chung như sau:

* Nó kết nối tới data warehouse
* Xử lý tiến trình mô hình hoá bằng việc biến đổi data từ bảng cũ sang bảng mới ở ngay trong data warehouse
* Nó sử dụng SQL để hình thành các biến đổi
* Cho phép người dùng chú thích, quản lý, kiểm tra thay đổi trên data models bất kì lúc nào
* Cho phép người dùng theo dõi lịch sử của các biến đổi dư liệu chỉ bằng một công cụ độc lập

=> *A data modeling layer* is *a system* containing the mapping between business logic and underlying data storage rules of your business. It exists primarily in the ELT paradigm, where data is loaded into the data warehouse first before being transformed. In this context, *data modeling* is the process of building and maintaining this layer.

- Thông thường data modeling layer sau này sẽ được kết nối tới một số công cụ phác hoạ hay BI layer => Người dùng không có kiến thức chuyên sâu vẫn có thể đăng nhập, tương tác với giao diện người dùng, lấy ra các phân tích họ cần mà không phải nhờ sự người khác có kiến thức.

=> Một data modeling layer tốt và được duy trì sẽ rất có ích vì:

* Business users có thể tự làm công việc phân tích theo mong muốn của mình
* DA sẽ tập trung được vào công việc chính của họ: duy trì các data pipeline và model layer
* Nếu DA có bận hay nghỉ việc thì các tri thức đã được ghi chú và tổ chức và không có rủi ro bị thất thoát

### Data Modeling Layer Concepts

#### Concept: Data Model

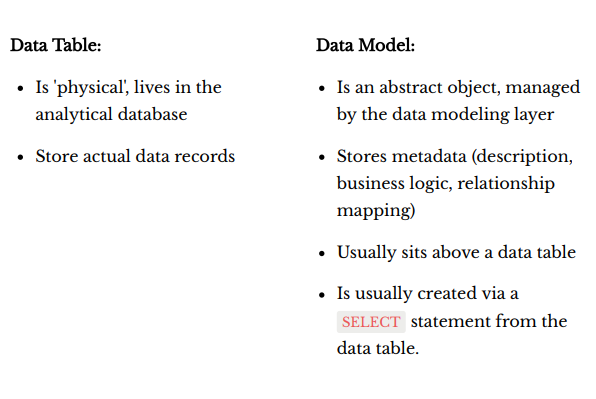
- Khi tiến hành thao tác dữ liệu ở trong data modeling layer, chúng ta không nên thực hiện trực tiếp đối với underlaying data table, thay vào đó hãy tạo một abstract object để có thể thao tác dễ dàng hơn => data model

- Data model là một abstract view ở trên cùng của một database table vật lý => thao tác không làm ảnh hưởng trực tiếp đến underlying data. Hầu hết các data modeling layer cho phép mọi người có thể lưu trữ thêm additional metadata mà có thể làm phong phú cho underlying data trong data table

=> Các dạng chính của metadata ở trong một data model có thể là textual desciptions, calculated dimensions, housekeeping metadata (means full active history of user accounts who have modified the model)

=> Data table quản lý bởi database, data model quản lý bởi data modeling layer mà bạn sử dụng

=> So sánh nhanh:



#### Concept: Relationship mapping

- Một ánh xạ quan hệ là một foreign relationship giữa 2 data model => tương tự với định nghĩa khoá ngoài trong csdl quan hệ

=> sẽ rất tốt nếu có thể tạo ra model bắt nguồn từ các model khác thay vì chỉ bắt nguồn từ các underlying database tables

#### Concept: Custom Field Logic

- Data modeling layers cho phép có thể tạo ra các công thức tính toán bao gồm từ các cột khác ở trong model => giống với concept ở trong BI tools – lưu ý rằng bạn có khả năng tăng các chiều đã tồn tại ở trong underlying tables

#### Concept: Models Builts On Top of Other Models

- Data model cho phép chú thích và tạo ra các derived field mới không ảnh hưởng đến các table quản lý chúng

#### Model Persistence (equivalent to materialized view)

- Khi data models thường là view on top của physical tables, gần như tất cả data modeling layer đều cho phép có thể duy trì các model này như những bảng mới ở trong data warehouse

### Putting things together

=> Những biểu đồ phụ thuộc này của models có thể coi là đại diện cho những gì một người phân tích sẽ làm hàng ngày

=> DA với công việc quan trọng nhất là tiến trình mô hình hoá các bảng thô sang các base model và rồi mô hình hoá các base model này sang các transformed model

=> Họ sẽ làm việc đó cho đến khi những data model có thể đại diện cho tất cả mọi thứ mà bên business cần

### What do business users get in the end?

=> Sử dụng UI, đằng sau đó là các công cụ, các lựa chọn sẽ được truyền thành SQL query tương ứng sau đó sẽ được gửi tới analytical database => gửi lại kết quả hiển thị cho người dùng

## Kimball’s Dimensional Data Modeling

### Why Kimball?

- Ý tưởng này (giới thiệu 1996) có độ phủ lan rộng nhất và được lựa chọn trong giới phân tích chuyên nghiệp => tập trung vào lược đồ không chuẩn hoá

- Bên cạnh đó còn có phương pháp Bill Inmon (giới thiệu 1990) => Tập trung vào lược đồ chuẩn hoá

- Hướng tiếp cận mô hình hoá dữ liệu thứ 3 là Data Vault (giới thiệu 2000s)

=> Các cách tiếp cận đều có giá trị riêng của mình nhưng tất cả đều cần phải update khi mà công nghệ về data warehouse phát triển nhanh

### The Star Schema

- Star schema là một con đường cụ thể để có thể tổ chức dữ liệu cho mục đích phân tích, bao gồm 2 dạng bảng:

* Fact table: hoạt động như một primary table cho schema. 1 bảng fact chứa các primary measurements, metrics hay ‘facts’ của một quy trình kinh doanh
* Các dimension table liên kết với fact table. Mỗi bảng dimension chứa các chiều hay là các thuộc tính đặc tả của fact table

=> Các bảng dimension sẽ xoay quanh bảng fact

### Kimball’s Four Step Process

- The star schema có những đặc tính sau:

* Flexible => Dữ liệu dễ dàng được chia nhỏ và phân vùng theo cách mà business user cần
* Extensible => để thích ứng với các thay đổi kinh doanh
* Performant (Chạy tốt như kì vọng) => phương pháp mô hình theo chiều của Kimball được phát triển khi hầu hết các hệ thống phân tích chạy trên relational database management systems a.k.a RDBMSes => Star schema chạy rất ổn trên hệ thống này khi các truy vấn được khai thác sử dụng ‘star join’ => Có thể hiểu chúng đều là tích đề-các của các bảng dimension

- Star schema chỉ thực sự hữu ích nếu có thể dễ dàng được áp dụng ở trong các doanh nghiệp => Để có thể bắt đầu star schema cho một công ty bất kì thì cần làm theo 4 bước sau:

1, Pick a business process to model

2, Decide on the grain

=> Grain có nghĩa là level của dữ liệu được lưu trữ trong bảng fact và chúng nên được phân chia thành level thấp nhất có thể (dữ liệu này không thể chia nhỏ ra được nữa mà vẫn có ý nghĩa) => Nếu không thì sẽ phải dùng ETL một lần nữa nếu data không có trong cái form truy vấn ở trong data warehouse => it is best to model at the lowest level possible from the beginning

3, Choose the dimensions that apply to each fact table row

4, Identify the numeric facts that will populate each fact table row

### Why Was Kimball’s Approach Needed?

1, It gave us speed to business value

2, Performnace reasons

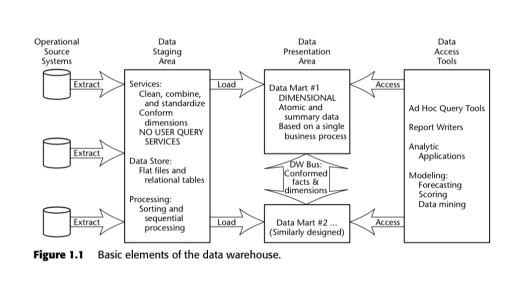
### Kimball-Style Data Modeling, Then And Now

- Sự khác biệt lớn nhất đang thay đổi đó là sự khác nhau về chi phí giữa data labor và data infranstructure

- Việc mô hình hoá dữ liệu theo Kimbal yêu cầu:

* Dành thời gian để thiết kế lược đồ trước
* Dành thời gian để xây dựng và duy trì các luồng dữ liệu cho việc khai thác các lược đồ
* Có một team được train về phương pháp Kimball để có thể tính toán, mở rộng và biến đổi những mô hình đang có để phản hồi lại những thay đổi trong quy trình kinh doanh

- Một typical data warehouse implementation project sẽ được mô tả như sau:



Có 2 ví dụ có thể áp dụng vào công việc:

1, Inventory Management

- Kimball mô tả cách để theo dõi việc di chuyển inventory là hoạt động kinh doanh phổ biến. Tuy nhiên luy ý rằng một bảng fact chứa quá nhiều các bước chuyển độc lập sẽ tạo nên một lượng khổng lồ để có thể phân tích được một cách tốt nhất

=> giải pháp chính là sử dụng ETL tools để tạo các ‘snapshot’ fact table có thể tập hợp các chi tiết về việc dịch chuyển inventory trong một khoảng thời gian cụ thể

=> DA chỉ cần sử dụng các bảng snapshot này và chỉ cần thực hiện một vài truy vấn ở trên bảng fact của inventory, bởi việc chạy toàn bộ bảng là một điều cực khó nhằn

- Ngày nay có 3 yêu cầu cho việc tạo các ‘snapshotting’ này:

* Các cloud data warehouse hiện đại thường được xây dựng ở phía sau là các kiến trúc dữ liệu theo cột => có thể quét dữ liệu hàng triệu dòng trong vài giây
* Hầu hết các cloud data warehouse hiện đại chạy trên kiến trúc xử lý song song khổng lồ => data warehouse có thể kiểm soát số lượng server chạy để chạy các truy vấn
* Cloud data warehouse sẽ được trả chi phí theo những gì bạn sử dụng

2, Slowly Changing Dimensions

- Có 3 giải pháp cho vấn đề này:

* Type 1: update các dimension column một cách ngây thơ => có vấn đề, càng sửa càng sai
* Type 2: Thêm một hàng mới, update key
* Type 3: Thêm một cột mới, là cột chứa thông tin cũ => hỗ trợ khả năng có thể có góc nhìn thực tế thay thế của những data giống nhau

- Hướng tiếp cận có những ưu điểm sau:

* Hướng tiếp cận này đảm bản sự câm bằng giữa computational resources và engineering time
* Dimensional data nhỏ và đơn giản nếu đem đi so sánh với fact data
* Snapshots giúp người phân tích có một mental model dễ dàng

### Applying Kimbal Style Dimensional Modeling to the Data Infranstructure of Today

#### Kimbal-style dimensional modeling is effective

#### Model As And When You Need To

- Với khả năng của máy tính hiện nay cho phép bạn tăng sự linh hoạt với công việc mô hình hoá => Bạn chỉ cần mô hình hoá khi bạn cần phải làm.

- Bạn sẽ tạo reports từ raw data tables trong hệ thống nguồn, mô hình hoá các bảng để phù hợp với kết cấu kinh doanh quan trọng nhất đối với users mà không cần để ý quá nhiều đến sự linh hoạt trong tương lai => Khi và chỉ khi việc những yêu cầu khó có thể đáp ứng thì bạn mới phải redo models dưới dạng một forrmal dimensional modeling manner.

=> Việc biến đổi trở nên dễ dàng khi được xử lý bên trong data warehouse, khi mà mô hình ELT sẽ thực hiện chúng => Khi mà mọi thứ được lưu trữ trong một data warehouse hiện đại thì chúng ta có thể thay đổi hướng tiếp cận mô hình bất kì lúc nào

=> Việc này còn trở nên dễ dàng hơn khi dùng cùng với các tool được thiết kế cho mô hình này => Chúng cung cấp các cấu trúc phù hợp và hỗ trợ về quản trị khi tạo, cập nhật và duy trì một data model mới đồng thời cho phép bạn có thể cập nhật mô hình theo hướng phát triển gia tăng

=> Việc phải xử lý toàn công việc mô hình hoá dữ liệu ngay khi bắt đầu một dữ án data warehouse không còn quá cần thiết

#### Use Technology To Replace Labor Whenever Possible

- Nguyên tắc chung là dùng công nghệ để thay thế các công việc của con người ngay khi có thể

- Phương pháp đương đại này phụ thuộc vào sức mạnh của cơ sở vật chất về dữ liệu hiện đại để có thể tiến hành xử lý các hoạt động thủ công không liên quan

- Với inventory modeling => sức mạnh của MPP columnar data warehouses khiến cho việc bỏ qua các bảng tổng hợp có thể bỏ qua, trừ khi chúng ta bắt buộc phải cần

- Với slowly changing dimensions => một phương pháp đã được sử dụng ở một số những công ty công nghệ hàng đầu => sử dụng việc phân hạch bảng để snapshot dữ liệu nhiều chiều trong toàn bộ thời gian => Không cần phải cài đặt một trong ba cách phản hồi

=> Ở trong cả hai trường hợp thì ý tưởng đều là ước lượng sự cân bằng giữa chi phí tính toán của máy tính và chi phí nhân công

## Modeling Example: A Real-world Use Case

### The Problem

### Creating The Pageview Model

### Evolving The Model to a Different Grain

### Exposing Self-service Analytics to Business Users

- Kimball đã đúng khi cho rằng data modeling là cách để giúp business user có thể điều hướng data ở trong data warehouse => Data modeling rất cần thiết để business users có cảm giác tốt khi sử dụng data

- Không cần phải dành nhiều thời gian để viết reports mới cho business users => Cần tái sử dụng các reports

### Takeaways

#### Let usage determine Modeling, not the reverse

- Vì toàn bộ raw analytical data được lưu ở trong data warehouse giống nhau nên sẽ không phải lo việc mất đi dữ liệu cần cho sự thay đổi trong tương lai => Có thể redo các models ở trong data modeling layer

#### Model just enough, but no more

=> Với những công ty có sự thay đổi nhanh thì deliver đủ business insights trong ngày là được

- Khi đến thời gian cần tạo ra session data model, có thể sẽ cần khoảng 2 ngày đến có thể tạo ra các truy vấn SQL và hiện thực hoá chúng trong các tool, sau đó một thêm một vài ngày nữa để có tạo ra các report cho các data model mới

=> Use such speed to your advantage. Chỉ mô hình hoá những gì bạn phải làm

#### Embed business logic in Data models, not Queries

- Hầu hết các data modeling layer tool đều khuyến khích bạn sẽ xây dựng các công thức tính toán và nghiệp vụ ở trong data model của bạn => làm cho các queries trở nên đơn giản và tránh các lỗi sinh ra bởi con người

#### The goal of modeling is self service

- Mục đích cuối cùng của việc mô hình hoá là tự phục vụ

- Self-service là quan trọng bởi vì điều đó có nghĩa là công ty của bạn không còn quá phụ thuộc vào data team

### Going Forward

- Sử dụng data modeling layer tool và ELT, những gì nhận được sẽ là sự linh hoạt, một hướng tiếp cận dễ dàng cho việc data modeling

# Using Data

## Data Servicing – a tale of three jobs

- Với mục đích đem đến góc nhìn tổng quan về analytics stack thì chương 4 sẽ đem đến những nội dung chính sau:

* Giải thích các giai đoạn thay đổi đã xảy ra trong 3 thập kỉ trở lại gần đây => giúp định hướng và tránh phải cảm giác bị mất kiến thức
* Đem đến góc nhìn toàn cảnh về thị trường của business intelligence reporting => giúp chọn ra các công cụ phù hợp
* Đem đến cảm quan về sự phát triển của reporting requirements sẽ được thấy ở trong một công ty

### A tale of three jobs

#### First job: Cognos and the ethernal wait

- Một ngày làm việc bình thường sẽ diễn RA theo workflow sau:

=> Sếp yêu cầu một vài thông tin số liệu

=> DA truy cập vào hệ thống Cognos (at the time, one of the most dominant business intelligence tools on the market) để kiểm tra liệu PowerPlay cubes có đang khả dụng

=> They point their own PowerPlay client to the cubes they want on the bank’s internal Cognos server

=> Slice and dice data within the cube to extract thê desired numbers for boss (form of Excel)

- Vấn đề xảy ra đó là nếu như sếp hỏi về các số liệu không được truy cập => sẽ phải bắt đầu một quá trình cực kì mệt mỏi như sau:

1, Verify that none of the cubes can access to contain the number that he needed to generate the report for his boss

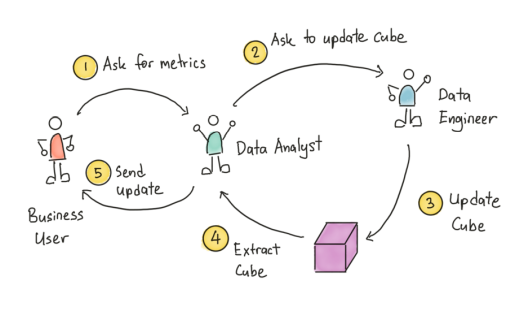
2, Contact the IT Services department with a work order => sent to a central enterprise resource planning system (count as an instance of *inter-department resource usage*)

3, Wait 3 weeks

4, New data was in Cognos system => wait a few hours for the data to be refreshed in order to build a new PowerPlay cube

5, If have many mistakes => return to step 2 and start over

=> Mô tả quá trình trên:



- Nó còn trở nên tệ hơn nếu mình phải gặp sự hỗ trợ từ dim-witted DE (gà mờ) hay bị lấy hết quyền ưu tiên khi có sự can thiệp từ sếp tổng có yêu cầu riêng

#### Second job: the metrics knife fight

- Tableau là một phần mềm BI tuyệt vời vào hoạt động theo hướng sau:

=> Kéo data từ SQL database và ném chúng lên Tableau đã được cài đặt trong máy của bạn

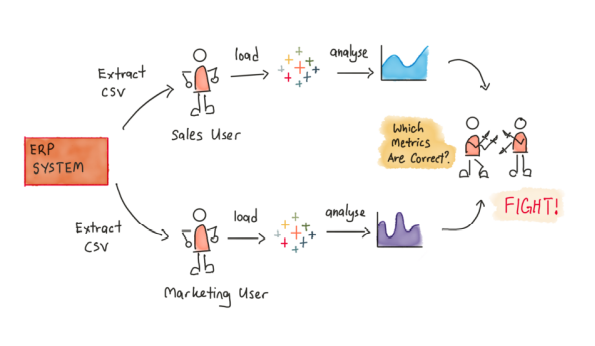
=> Kéo Excel về và ném chúng lên Tableau

=> Kéo file CSV lên vào ném chúng lên Tableau

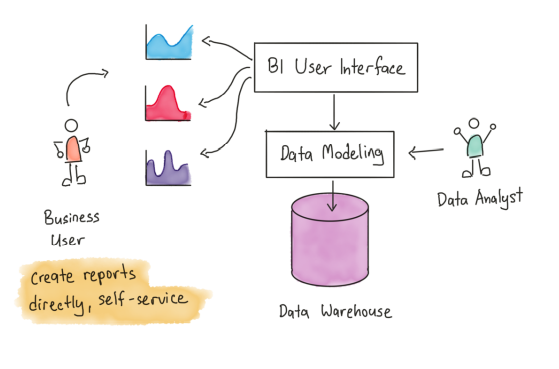
=> Connect Tableau với 1 OLAP cube hoặc trực tiếp tới SQL database

- Tableau là một công cụ drag-and-drop, và đến thời điểm này nó là một công cụ trực quan rất tốt mà business users có thể học cách sử dụng để có thể tự tạo ra những reports họ cần

- Tuy nhiên đã xuất hiện vấn đề với hướng tiếp cận này, khi các phòng ban khác nhau định dạng và tạo dạng các bảng dữ liệu khác nhau từ những metrics giống nhau => mâu thuẫn xảy ra => Ban hành quy định rằng các định nghĩa về số liệu phải được lưu trữ tại một địa điểm trung tâm, được duy trì bởi một team BI



#### Third job: the data modeling layer



- Không giống như trong mô hình Tableau, logic kinh doanh được viếtmột lần - bởi nhóm dữ liệu - trong lớp mô hình dữ liệu. Các mô hình này sau đó được kết hợp lại bởi các nhà phân tích khác và bởi những người dùng kinh doanh không chuyên về kỹ thuật để tạo ra các báo cáo và giao diện khám phá mà doanh nghiệp cần

### Frank Bien’s Three Waves

- Năm 2017, CEO Looker Frank Bien cho ra mắt cuốn sách có tựa đề “Catching the Third Wave of Business Intelligence” => mô tả chính xác 3 đợt sóng đã minh hoạ ở trên a.k.a 3 lần thay đổi về cách tiếp cận và công nghệ áp dụng

- Trong thập kỉ trước, 2 công nghệ đột phá đã thay đổi landscape một lần nữa:

* MPP data warehouses
* Columnar datastores đã có thể match với OLAP cubes ở trong analytical performance

=> MPP data warehouses thường đi kèm với một phát minh khác là cloud vendor pricing model => Doanh nghiệp ngày nay chỉ cần trả phí cho dung lượng và sức mạnh tính họ sử dụng

- Modern data warehouses ngày càng trở nên đủ rẻ và đủ mạnh để có thể tự vận hành

### The Major takeaways

- Việc có thể nắm bắt ý tưởng của một công cụ BI có thể được áp dụng vào nhưu thế nào quan trọng việc lựa chọn các công cụ => Có 3 xu hướng xuất hiện

1, Phương pháp tiếp cận trong BI tools bị giới hạn bởi công nghệ hiện có

2, Tiếp cận trong BI thường sẽ xử lý những vấn đề của công cụ trong thế hệ trước

3, Các thế hệ sẽ cùng tồn tại và gắn liền với các công ty trong khoảng thời gian dài

## Navigating the business intelligence tool space

Thị trường các công cụ BI ngày càng trở nên dễ bị nhầm lẫn và rắc rối khi lựa chọn vì các tool từ các mô hình cũ vẫn sẽ tồn tại trong khoảng thời gian dài. Dựa vào lịch sử phát triển chúng ta sẽ cùng bàn bạc để có những so sánh về các công cụ

### SQL vs Non-SQL

- Một số BI tool yêu cầu kiến thức về SQL nhưng một số thì không

- Các BI tool vận hành on top của OLAP cubes có xu hướng không sử dụng SQL

=> Microsoft’s MDX là một ngôn ngữ được phát triển cho việc vận hành 1 data cube. Rất nhiều các bên kinh doanh OLAP cube cài đặt ngôn ngữ này và được xem xét có thể thay thế cho SQL

- Một số BI tool khác thì được định hướng sử dụng SQL

=> Ví dụ:

* Chartio comes with a sophisticated Visual SQL mode
* Holistics treats direct SQL query creation as a secondary access path
* Mode Analytics assumes that data extracted from your data warehouse should be used for both data science
* Looker requires all data analysts to write in LookML, which is then translated into SQL for the underlying database

#### The shift from non-SQL based analytics to SQL based analytics

- Trong hơn 5 năm trở lại đây, bước thay đổi hướng tới việc sử dụng SQL đã diễn ra

- Những trend nhằm tránh việc sử dụng SQL đều đi vào dĩ vãng và hầu hết các analytical datastore ngày nay đều được chuẩn hoá cùng với SQL

=> Kể cả non-SQL datastores như Spanner database từ Google hay Hadoop cũng thiết lập SQL như 1 ngôn ngữ truy vấn

=> Có một vài lý do cho việc chuẩn hoá cùng với SQL:

* Các người dùng SQL thành thạo có thể tận dụng sức mạnh của SQL cloud-based data warehouse để làm việc hiệu quả mà không cần phải học một ngôn ngữ hay công cụ độc quyền nào khác
* SQL là một text-based nên có thể được lưu trữ ở 1 version control system => dễ dàng quản lý
* Sự thống trị của SQL dẫn đến số lượng lớn các công cụ BI cài đặt SQL như là một ngôn ngữ giao diện phổ thông

- Trend này sẽ không mất đi sớm khi các chuẩn đều được yêu thích qua thời gian

### Embedded Datastore vs External Datastore

#### Sidebar: Massively Parallel Processing vs Relational Database?

- Thế nào có thể coi là một ‘modern data warehouse’? => Cần có 2 đặc điểm sau:

1, Đó là một column-oriented database

2, Có kiến trúc MPP

### In-Memory vs In-Database

- Các công cụ như Holistics, Redash, Chartio, Metabase và Looker chạy các truy vấn SQL trên một powerful database => Những công việc nặng nhất đều được tiến hành bởi chính database

- Các công cụ như Tableau hay PowerBI lại yêu cầu analyst kéo data khỏi central data warehouse và chạy các truy vấn phân tích ở máy riêng

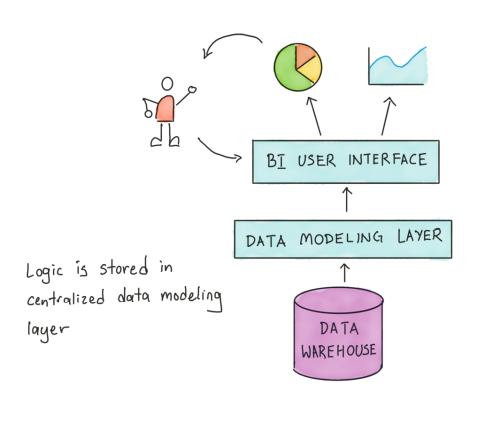
=> Khi đánh giá một công cụ, bạn sẽ cần hiểu rằng quy trình nào của công cụ bạn đang muốn sử dụng nó

### Modeling vs Non-modeling BI tools

- Trong khi làm việc, có nhiều các data team đương thời mà họ cài đặt kĩ thuật ELT sử dụng data modeling layers như dbt hay Dataform, hoặc sử dụng phương pháp ETL truyền thống như Pentaho hay Talend

- Holistics và Looker tạo nên sự khác biệt, khi họ cung cấp một modeling layer xuyên suốt chức năng BI => Toàn bộ logic sẽ được tập trung hoá ở data modeling layer, từ đó tăng việc gắn kết giữa các số liệu và khả năng tái sử dụng các logic đối với doanh nghiệp

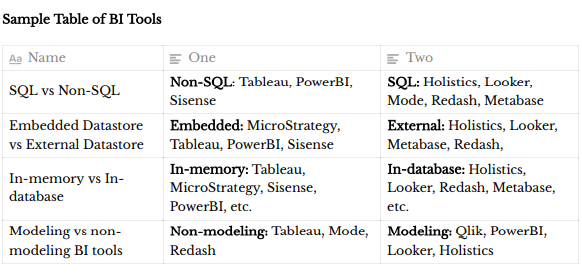
- Một lợi ích khác khi bạn sử dụng BI tool mà có modeling layer đó là maintaining context => Bạn có thể theo dõi được toàn bộ việc sửa đổi report data từ nguyên bản ban đầu, khi mà công cụ này nó chạy một phần trong mọi phép biến đổi



### Our biases, revisited

* Prefering ELT over ETL => ELT gives you the power of more flexible data modeling practices => You may choose to skip the up-front modeling cost and only choose to do when I receive the requirement
* Prefering using a cloud data warehouse => modern MPP data warehouse fundamental shift in capabilities
* Believing data modeling is essential => data modeling at a central layer enables organizational self-service
* SQL based analytics will win over non-SQL based analytics since the entire industry have standardlized on SQL in recent years
* Believing analytics workflow/operations are more important than a singular focus on visualizations

### Wrapping up



## The arc of adoption

- Sau khi có những sự phân loại cơ bản về thị trường BI tool thì bạn sẽ cần phải biến về quá trình thay đổi về các yêu cầu bạn nên nhận ra ở trong doanh nghiệp

### One: Ad-hoc queries

- Việc bạn phục vụ các truy vấn như thế nào phụ thuộc chủ yếu vào các công cụ có thể khả dụng cho bạn => Nếu có thể truy cập được vào centralized data warehouse thì bạn sẽ cần viết truy vấn SQL dưới dạng ad-hoc để sinh ra các con số cần. Nhưng nếu không, bạn sẽ phải tìm được đúng nguồn dữ liệu, lấy nó ra và phân tích ngay trên thiết bị cá nhân, và nếu có bất cứ sai số nào ở input thì sẽ phải lặp lại toàn bộ quá trình

### Two: Static reports and Dashboards

- Khi mà càng nhiều người có xu hướng sử dụng data để củng cố cho luận điểm của mình a.k.a khi doanh nghiệp mở rộng quy mô => Dẫn đến số lượng data request gửi tới data team là rất nhiều => Cần có giải pháp để làm giảm các yêu cầu giống nhau và có sự lặp lại

- Data leader bắt đầu sẽ dùng BI tool để tạo ra các dashboard cho những metric có thể dự đoán, từ đó giảm thời gian phải tiếp nhận các yêu cầu ad-hoc

### Three: Self-Service

- Data team bắt đầu lại rơi vào tình trạng quá tải, khi business operators bắt đầu đưa ra các form tinh vi hơn và cần nhiều ad-hoc, exploratory data request

- Tuỳ vào mô hình BI của doanh nghiệp mà hướng giải quyết sẽ khác nhau

=> Có thể sẽ training nhân viên học cách sử dụng SQL

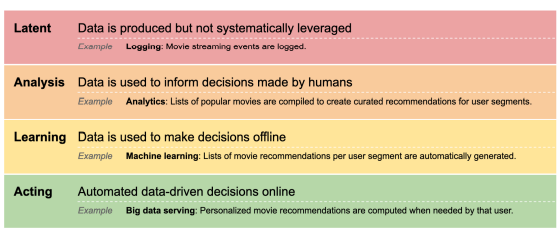
=> Mua các phần mềm như PowerBI hay Tableau Desktop

=>> Hai hướng tiếp cận này đều có vấn đề riêng nhưng vấn đề lớn nhất đó là chúng thường dẫn đến sự xung đột về output đầu ra, khi các business user khác nhau sẽ có định nghĩa khác nhau về số liệu trong các phân tích của họ => Gây mất kết nối

### The arc: Then and now

Ý tưởng của arc có thể hiểu giống như là ‘data maturity mode’

- Dưới đây là một ví dụ của Jon Bratseth:



- Một mô hình về phân tầng data mà chúng ta thường thấy hơn đó là của Emily Schario từ GitLab, có 3 level của một việc phân tích dữ liệu:

1, Reporting => Lowest level. If you do not collect data or if you do not have the cultural expectation of using it, you’re not going to base your decisions on facts

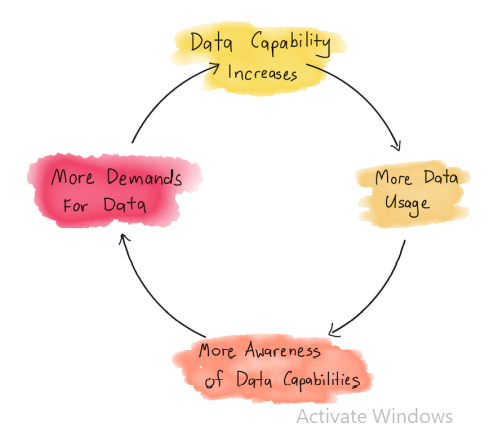
2, Insights => if reporting is about gathering facts to report on them, insights are about understanding relationships between facts

3, Predictions => begin to see sophisticated techniques lke statistical analysis and machine learning

=>> Vậy vì sao arc lại xuất hiện? The arc xuất hiện bởi data-driven thinking is a learned organizational behavior => nó sẽ lan truyền dần trong văn hoá của một công ty, với tư cách là một người trong team data thì bạn sẽ có thể theo dõi hiện tượng đó

=> khả năng lưu trữ dữ liệu mà bạn xây dựng cho team sẽ ảnh hưởng đến tốc độ lan truyền về tư duy data-driven thinking trong doanh nghiệp => *The more data capabilities you have, the more people will be exposed to the potential of using data to advance their arguments. The more data capabilities you build up, the more leverage you give to data-driven people in your company's culture to spread their way of thinking*

- Vòng tròn sẽ được lặp lại như sau:



### Solving the self-service problem today

- Có thể thấy rằng data modeling nằm tại một central data warehouse là giải pháp cho vấn đề

=> Define your business definitions once, in a modeling layer, and then parcel these models out for self-service

# Conclusion

## The End

- Toàn bộ hệ thống phân tích điều phải tiến hành 3 công việc cơ bản. Những công việc này sẽ cho chúng ta một framework hữu ích để có thể nghĩ về việc xây dựng cơ sở hàng tầng dữ liệu

=> 3 điều đó là:

1, Bạn sẽ phải collect, consolidate và store data ở trong một central data warehouse

2, Bạn phải xử lý data, bao gồm transform, clean up, aggregate và model data đã được đẩy lên một central data warehouse

3, Bạn phải present data, gồm có visualize, extract hay đẩy data lên các dịch vụ khác hoặc tới người dùng cần chúng

- Bên cạnh dó, các góc nhìn về những điều khác trong lĩnh vực này đã được nói đến:

* To examine the rise of the modern data warehouse as being the center of most contemporar analytics stacks
* To show how ELT provides powerful benefits that stretch beyond minor operational improvements
* To show the outline of a contemporary dimensional data modeling approach
* To demonstrate how to do that using a new breed of tools we've named "data modeling layer tools" and walking through a set of concepts that are common to those tools
* To sketch for the shape of the entire BI tool landscape

# HỎI ĐÁP

## Kho dữ liệu là gì?

- Data warehouse là một cơ sở dữ liệu phân tích có nhiệm vụ chính là lưu trữ dữ liệu và xử lý dữ liệu đó cho mục đích phân tích

- Chúng ta sẽ cần bởi có một data warehouse system bởi:

+ Dữ liệu có ở nhiều nguồn khác nhau => cần một nơi để có thể tập hợp chúng lại cho các công việc, mục đích sau này

+ Khi mà muốn chạy một truy vấn hay công việc phân tích nào đó, việc mà thực hiện chúng trên máy tính cá nhân với khối lượng dữ liệu như vậy sẽ lãng phí cả về thời gian và chi phí

- Data warehouse sẽ có 3 chức năng chính sau:

+ Lưu trữ dữ liệu từ nhiều nguồn

+ Xử lý dữ liệu theo hướng chuyên sâu

+ Dữ liệu sau khi được xử lý thì sẽ được visualize và được đem đến cho end user theo mong muốn của họ

## Quá trình ETL là gì?

- Quá trình ETL viết tắt của extract (trích xuất), transform (chuyển đổi), load (tải)

- Đây là quy trình chính trong một data warehouse system với cụ thể mục đích của 3 bước:

+ Đầu tiên, bạn sẽ phải trích xuất dữ liệu từ nhiều nguồn (chủ yếu là 3 nguồn gồm nguồn từ người dùng, từ bên thứ 3 và từ bên trong doanh nghiệp)

+ Sau khi trích xuất, các dữ liệu sẽ được biến đổi sao cho phù hợp với mục đích phân tích

+ Cuối cùng, các dữ liệu sau khi được xử lý sẽ được tải lên trên data warehouse

=> Các key thing ở đây là việc data transformation diễn ra bên ngoài data warehouse

## Data transformation là gì? Tại sao lại cần nó?

- Data transformation là quá trình biến đổi dữ liệu từ cấu trúc này sang một cấu trúc khác nhằm phục vụ mục đích phân tích

- Thường sẽ có 3 use case mà cần đến việc tranform data là

+ Tổng hợp dữ liệu

+ Dọn dẹp dữ liệu

+ Tính toán lại

- Data transformation có nhiều mục đích rất tốt trong quy trình ETL:

+ Việc tái sử dụng sẽ được tối ưu

+ Báo cáo và phân tích sẽ có sự nhất quán hơn

+ Tiết kiệm thời gian, tăng hiệu suất

+ Chi phí hợp lý

## Bảng dim, bảng fact là gì? Tại sao lại phải chuyển đổi thành 2 bảng này?

- Bảng dimension bao gồm các thuộc tính của một khía cạnh cụ thể nào đó trong một lĩnh vực, đi kèm thường sẽ có 1 primary key dùng được tham chiếu từ bảng fact

- Bảng fact có thể coi như một primary table, nó chứa có thông tin cơ bản của một data underlying table, bảng fact sẽ có các foreign key tham chiếu đến các primary key tương ứng của bảng dimension

- Việc chuyển đổi của 2 bảng này có ý nghĩa rất lớn đến việc chạy các truy vấn hay các quy trình xử lý trong data warehouse, đặc biệt là với hệ thống sử dụng dimensional data store để lưu trữ dữ liệu

## Các lược đồ sử dụng trong kho dữ liệu?

- Có 3 dạng lược đồ phổ biến đó là:

+ star schema

+ snowflake schema

+ galaxy schema

Trong đó thì mới đọc về star schema thôi

=> star schema là mô hình trong đó các bảng dim sẽ xoay xung quanh bảng fact => nếu sử dụng dimensional data store thì lược đồ này sẽ giúp công việc truy vấn trở nên đơn giản hơn và việc tải dữ liệu lên DDS cũng nhanh hơn