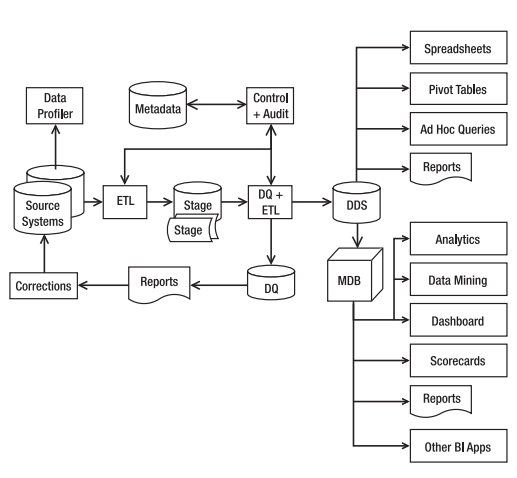
# Introduction to Data Warehousing

## What is a Data Warehouse?

- Data warehouse là một hệ thống retrive và arregrate data một cách định kì từ các hệ thống nguồn đến các kho dữ liệu theo chiều hoặc được chuẩn hoá

- Nó thường có lịch sử truy cứu trong nhiều năm và được dùng để truy vấn cho BI và các hoạt động phân tích khác

- Nó được cập nhật theo từng đợt



- Bên trên là một mô hình của data warehouse system và các ứng dụng đi kèm

- Với mô hình này:

* Hệ thống nguồn (*source systems*) là các hệ thống OLTP (Online Transaction Processing) chứa data bạn muốn tải chúng lên data warehouse. OLTP là một hệ thống mà mục đích chính là capture và store các business transaction
* Data của hệ thống nguồn được kiểm tra bằng cách sử dụng một *data profiler* để có thể hiểu được đặc điểm của data => *Data profiler* là công cụ có khoang chứa để phân tích data
* Hệt thống *ETL* đem data từ nhiều nguồn đến staging area => ETL có khu vực chứa để có thể kết nối hệ thống nguồn, đọc data, chuyển đổi data và tải chúng lên hệ thống đích (KHÔNG phải là data warehouse) => ETL sẽ tích hợp, chuyển đổi và tải data lên *dimensional data store (DDS)*
* *DDS* là cơ sở dư liệu chứa data của data warehouse nhưng ở nhiều format khác nhau hơn so với OLTP => Data ở đây được sắp xếp theo format về chiều sẽ phù hợp cho việc phân tích và được tích hợp từ nhiều hệ thống nguồn trước đó
* Khi ETL tải data lên DDS, *data quality rules* sẽ kiểm tra chất lượng của data. Những data xấu sẽ được đẩy đến *data quality (DQ)* database để báo cáo và sau đó được sửa lại trong hệ thống nguồn, nếu data chỉ sai trong giới hạn cho phép thì sẽ tự động được chấp nhận
* ETL được quản lý và dàn dựng bởi hệ thống điều khiển, dựa trên các hậu quả, quy luật và logic được lưu trữ trong *metadata* => *Metadata* là cơ sở dữ liệu chứa các thông tin về data structure, data meaning, data usage, data quality rules và các thông tin khác về data
* *Audit system* ghi nhật ký các hoạt động và việc sử dụng hệ thống vào cơ sở dữ liệu siêu dữ liệu => Là 1 phần của ETL điều hành các hoạt động vận hành của quy trình ETL và ghi lại các thống kê vận hành => Được dùng điều hiểu được những gì đã diễn ra trong quy trình ETL
* *Spreadsheets, pivot tables, reporting tools và SQL query tools* là các front-end tool dùng để lấy lại và phân tích data ở trong DDS
* Những ứng dụng trên được sử dụng thì data trong *DDS* sẽ được tải lên *multidimensional database (MDBs) a.k.a cubes*
* Các công cụ như *analytics applications, data mining, scorecards, dashboards, multidimensional reporting tools* và các công cụ BI khác có thể lấy lại dữ liệu một cách có tương tác từ *MDBs* => lấy lại data để có thể tạo ra nhiều features và results ở trên màn hình hiển thị để user có thể có cái nhìn cụ thể hơn về doanh nghiệp của họ

- Nhưng không phải lúc nào hệ thống data warehouse sẽ cần đầy đủ các thành phần kể trên, dưới đây là dạng đơn giản nhất mà chúng ta có thể xây dựng:



=> Chỉ cần có ETL và DDS là đủ, ngoài ra lưu ý rằng hệ thống nguồn không phải là một phần của hệ thống data warehouse

### Retrieves data

- Lấy lại dữ liệu được tiến hành hệ thống ETL, viết tắt của extract, transform và load

Việc chuyển đổi được sử dụng để thay đổi data sao cho phù hợp với format và tiêu chuẩn của hệ thống đích; để lấy được các giá trị mới đã được tải lên hệ thống đích; hoặc xác nhận data từ hệ thống nguồn

- ETL không chỉ được dùng để tải dữ liệu lên data warehouse mà còn được dùng cho nhiều dạng dịch chuẩn dữ liệu khác

- Hầu hết các hệ thống ETL đều có cơ chế dọn dẹp data từ hệ thống nguồn trước khi đẩy lên warehouse. Data cleansing là quá trình xác định và làm đúng lại các dữ liệu xấu => cài đặt data quality rules để xác định đâu là dirty data

=> Sau khi được trích xuất và trước được tải lên warehouse thì data sẽ được kiểm tra

* Nếu ok => load lên warehouse
* Nếu không ok => có 3 options: có thể bị bỏ đi (rejected), được cho là đúng (corrected) hoặc được phép tải lên warehouse => phụ thuộc vào tình huống, level rủi ro, dạng quy định…

- Một hướng tiếp cận khác đó là ETL đó là ELT

=> Với hướng này, data sau khi được trích xuất sẽ được tải lên data warehouse ở dạng thô, toàn bộ quy trình biến đổi, xử lý sẽ diễn ra bên trong data warehouse

=> Hướng này được thiết kế cài đặt để có thể tận dụng ưu điểm sức mạnh của data warehouse data base như hệ thống MPP

### Consolidates data

- Một data warehouse tập hợp rất nhiều transactional system

- Hãy đi vào tìm hiểu các định nghĩa sau:

#### Data availability:

- Khi tập hợp data từ nhiều hệ thống nguồn, rất có thể một phần data khả dụng cho hệ thống này nhưng lại không đối với hệ thống khác

- Khi tập hợp các data đến với các hệ thống khác nhau thì bạn cần có lưu ý đến các cột không khả dụng và các level bị thiết trong cây thừa kế

#### Time ranges:

- Một khối dữ liệu giống nhau tồn tại ở các hệ thống khác nhau nhưng lại có khoảng thời gian khác nhau

- Bạn luôn phải kiểm tra xem khoản thời gian nào có thể áp dụng cho dữ liệu kia trước khi tập hợp chúng => Nếu không bạn sẽ gặp rủi ro rằng dữ liệu ở trong data warehouse sẽ bị sai

#### Definitions:

- Đôi lúc dữ liệu giống nhau nhưng là chứa các thông tin khác nhau

- Bạn cần phải kiểm tra định nghĩa của từng thành phần trong dữ liệu => Chúng có trùng tên KHÔNG có nghĩa là chúng giống nhau

#### Conversion:

- Khi tập hợp dữ liệu đến với các hệ thống nguồn khác nhau, thi thoảng bạn sẽ cần thực hiện việc quy đổi (conversion) bởi data trong hệ thống nguồn ở một dạng khác => nếu thêm vào mà không convert trước đó, data trong warehouse sẽ bị sai

- Một số trường hợp việc thay đổi tỉ lệ quy đổi là liên tục => cần phải biết được sao bao lâu thì cần phải tiến hành việc quy đổi

#### Matchings:

- Việc nối (matching) là một quá trình xác định xem khối dữ liệu ở hệ thống này có giống với khối dữ liệu ở hệ thống kia

- Việc nối rất quan trọng bởi nếu bạn nối data sai thì bạn sẽ tạo ra các dữ liệu sai ở trong data warehouse

### Periodically

- Việc lấy lại dữ liệu và tập hợp dữ liệu không phải chỉ thực hiện đúng một lần, chúng diễn ra nhiều lần và thường với khoảng thời gian định kì => nếu chỉ thực hiện một lần thì data sẽ trở nên lỗi thời

- Bạn có thể xác định khoảng thời gian hợp lý cho hai công việc trên phụ thuộc vào yêu cầu doanh nghiệp và tần suất của việc cập nhật dữ liệu ở trong hệ thống nguồn => Khoảng thời gian tiến lấy lại dữ liệu cần phải giống với tần suất cập nhật dữ liệu của hệ thống nguồn

- Bên cạnh đó, bạn cũng cần đảm bảo rằng khoảng thời gian tiến hành khối phục dữ liệu sẽ phù hợp với yêu cầu của doanh nghiệp

### Dimensional data store

- Một DDS là một hay vài cơ sở dữ liệu containing a collection of dimensional data marts

- Một dimensional data mart là một nhóm các related fact table và dimensional table tương ứng chứa các định lượng, đo lường của business events được phân thành các danh mục bởi dimensions của họ

- Một DDS sẽ ở dạng không chuẩn hoá và dimensions are conformed => có nghĩa là chúng sẽ là những dimensional table giống nhau hoặc là một cái là bảng con của cái còn lại

- Một DDS có thể được cài đặt một cách vật lý từ một vài lược đồ khác nhau => Có 3 dạng biểu đồ thường được sử dụng đối với DDS đó là star schema, snowflake schema và galaxy schema, trong đó:

* Với star schema, một chiều sẽ không có bảng con
* Với snowflake schema, một chiều sẽ có bảng con hay chiều con => mục đích của việc có bảng con đó là làm giảm tối đa dữ liệu dư thừa
* Với galaxy schema thì bạn sẽ có 2 hoặc nhiều hơn 2 bản fact xung quanh các bảng dim giống nhau

- Lợi ích của việc sử dụng star schema đó là nó đơn giản hơn 2 lược đồ còn lại và giúp cho quá trình ETL dễ dàng trong việc load data lên DDS

- Lợi ích của việc sử dụng snowflake schema là giúp tối ưu một số ứng dụng phân tích và giảm dữ liệu dư thừa và không gian lưu trữ

- Lợi ích của việc dùng galaxy schema đó là khả năng có thể mô hình các business event một cách chính xác khi sử dụng các bảng fact

### Normalized data store

- Một NDS là một hay nhiều cơ sở dữ liệu quan hệ không có hoặc có ít các dữ liệu dư thừa

- Một cơ sở dữ liệu quan hệ là csdl chứa các bảng thực thể có quan hệ cha con giữa chúng

- Sự chuẩn hoá (Normalization) là quá trình loại bỏ dư thừa dữ liệu bằng cách cài đặt các quy luật chuẩn hoá => Có 5 bậc của normal form từ first đến fifth. Một NDS thường là third normal hoặc cao hơn là fourth hoặc fifth

- Một DDS phù hợp hơn cho việc lưu trữ dữ liệu trong warehouse cho mục đích truy vấn và phân tích data hơn là một NDS => Bởi nó đơn giản hơn và cho hiệu năng truy vấn cao hơn

- Một NDS phù hợp cho việc tích hợp data từ nhiều hệ thống nguồn khác nhau, đặc biệt khi ở third normal form hoặc cao hơn => Bởi chỉ có một nơi mà có thể update mà không có dư thừa dữ liệu => thường được sử dụng cho các data warehouse của doanh nghiệp, data từ đây được tải lên DDS cho việc truy vấn và phân tích

- Một vài ứng dụng chạy trên một DDS, đó là một csdl quan hệ gồm các bảng có chiều và cột, một vài ứng dụng trong csdl đa chiều bao gồm các cube với cells và dimensions

### History

- Một sự khác biệt rõ ràng giữa transactional system và data warehouse system đó là capabilty và capacity để lưu trữ lịch sử

- Hầu hết transactional system chỉ lưu trữ một phần lịch sử (~ 3 năm) còn data warehouse system lưu trữ lịch sử một thời gian dài

- Một data warehouse lưu trữ lịch sử rất lâu trong active system, phụ thuộc vào yêu cầu từ phía doanh nghiệp

- Một bảng data warehouse có thể sẽ rất lớn, điều này cùng xảy ra trong ngành công nghiệp viễn thông và bán lẻ online, đặc biệt khi bạn lưu trữ việc truy cập webpage ở trong data warehouse

=> It is important for a data warehouse system to be able to update a huge table bit by bit, query it bit by bit, and back it up bit by bit

- Các tính năng của csdl như phân hạch bảng và truy vấn song song có thể sẽ hữu ích cho một hệ thống data warehouse:

* Phân hạch bảng là phương thức chia các bảng bởi các dòng thành các phần nhỏ và lưu trữ từng phần trong các file khác nhau để tăng việc tải dữ liệu hiệu năng truy vấn
* Truy vấn song song là quá trình các truy vấn độc lập được chia thành các phần nhỏ hơn và mỗi phần được đưa tới các module xử lý truy vấn độc lập

=> Kết quả truy vấn từ mỗi module sẽ được combine và gửi lại lên ứng dụng front-end

- Một data warehouse lưu trữ lịch sử của master data là một tính năng đặc biệt => đó là slowly changing dimension (SCD) => Là một kĩ nghệ được sử dụng trong dimensional modeling cho việc bảo tồn thông tin về lịch sử của dimensional data => có 3 dạng SCD chính:

* Dạng 1: Sửa lại thông tin, đồng nghĩa với việc không giữ lại lịch sử
* Dạng 2: Giữ lại thông tin bằng cách tạo thêm dòng
* Dạng 3: Giữ lại thông tin bằng cách tạo thêm bảng

- Cũng liên quan đến lịch sử, data warehouse lưu trữ các snapshot định kì của hệ thống nguồn vận hành. Một snapshot là 1 hay nhiều master table được lấy ở một thời điểm cụ thể => periodic snapshot là một snapshot sẽ được tạo ra ở khoảng thời gian định kì

### Query

- Truy vấn là quá trình lấy data từ data store để làm hài lòng những tiêu chuẩn cụ thể

- Một data warehouse sẽ được xây dựng cho việc truy vấn => là mục đích đầu tiên cho việc tồn tại của nó

- User không được phép cập nhật data warehouse mà chỉ có thể truy vấn data warehouse => Chỉ ETL được phép cập nhật data warehouse (1 key difference giữa transactional system và data warehouse system)

- Với mục đích chỉ là các truy vấn và báo cáo đơn giản thì bạn có thể truy vấn vào hệ thống nguồn một cách trực tiếp, Nhưng nếu tiến hành các phân tích yêu cầu hiệu năng cao như lợi nhuận khách hàng, phân tích dự đoán, viễn cảnh “nếu như”, các hoạt động phân tích slice and dice… thì rất khó để làm trên hệ thống nguồn

=> Lý do là source system thường là transactional system và được sử dụng bởi rất nhiều người dùng. Một tính năng quan trọng của transactional system đó là khả năng cho phép nhiều người dùng cập nhật và select từ hệ thống cùng lúc. Để làm được điều đó, thì phải có khả năng thực hiện rất nhiều database transactions trong một thời gian ngắn và rất nhanh, bên cạnh đó việc cập nhật data và sự nhất quán của data cũng phải được quan tâm

- Thực hiện các truy vấn phức tạp trong database chuẩn hoá sẽ chậm hơn là trong database không được chuẩn hoá (ví dụ là data warehouse)

- Lý do thứ 2 bạn không nên truy vấn hệ thống nguồn một cách trực tiếp đó là vì một công ty có thể có nhiều hệ thống nguồn hay front-office transactional systems => Khi truy vấn một hệ thống nguồn thì bạn chỉ có được một phần của data => Với data warehouse thì lại tập hợp data từ nhiều hệ thống nguồn và khi truy vấn vào đây thì data của bạn đã được tích hợp

### Business Intelligence

- BI là một tập các hoạt động để có thể hiểu các tình huống trong doanh nghiệp bằng việc thực hiện rất nhiều các dạng phân tích trên data của công ty cũng như data bên ngoài từ bên thứ bạn đế giúp đưa ra các quyết định liên quan đến vận hoành doanh nghiệp có chiến lược và chiến thuật và tiến hành các hoạt động cần thiết cho việc cải thiện hiệu quả kinh doanh

- Nó bao gồm gathering, analyzing, understanding và manageing data về hiệu quả vận hành, khách hàng, hoạt động nhà cung cấp, hiệu quả tài chính, thay đổi thị trường, cạnh tranh…

- Một ví dụ về BI sẽ có nững công đoạn sau:

* Business performance management, bao gồm các việc tạo ra các biểu thị hiệu suất chính chỉ doanh số hàng ngày, tận dụng nguồn lực và chi phí vận hành chính của mỗi vùng… => Cho phép mọi người có thể tiến hành các hoạt động chiến thuật để đạt được hiệu quả vận hành trong các track mong muốn
* Customer profitability analysis => xác định chi phí một cách chính xác nhất đối với những phần nhỏ nhất của business transaction hay có thể coi là chi phí dựa trên hoạt động
* Statistical analysis such as purchase likelihood or basket analysis => Basket analysis là quá trình phân tích dữ liệu doanh số để xác định sản phẩm nào nên được mua và đặt hàng cùng lúc => Likelihood được thể hiện trong việc đo lường thống kê như support and confidence level => Ứng dụng vào ngành công nghiệp chế tạo và bán lẻ nhưng cũng được ứng dụng một phần vào ngành dịch vụ tài chính (like bank…)
* Predictive analysis => dự báo doanh số, doanh thu cho việc lên kế hoạch về ngân sách của năm sau và thêm vào một số nhân tố khác như tăng trưởng hữu cơ, các tình huống xảy ra của nền kinh tế và hướng phát triển của doanh nghiệp

- According to the depth of analysis and level of complexity => Chia các hoạt dộng BI thành 3 nhóm:

• Reporting, such as key performance indicators, global sales figures by business unit and service codes, worldwide customer accounts, consolidated delivery status, and resource utilization rates across different branches in many countries

• OLAP, such as aggregation, drill down, slice and dice, and drill across

• Data mining, such as data characterization, data discrimination, association analysis, classification, clustering, prediction, trend analysis, deviation analysis, and similarity analysis

#### Reporting

- Report là một program mà lấy data từ data warehouse và hiện thị chúng đến user

- Các báo cáo được xây dựng dựa theo sự cụ thể về chức năng, hiện thị các DDS data được yêu cầu bởi business user để phân tích và hiệu được tính huống kinh doanh

- Các form chủ yếu của report sẽ là tabular form chức các cột độc lập, ngoài ra có cross tab và matrix

- Data warehouse report được sử dụng để trình bày các dữ liệu kinh doanh đến users nhưng cũng được sử dụng cho mục đích quản trị data warehouse

#### Online Analytical Processing (OLAP)

- OLAP là một hoạt động phân tích một cách có tương tác các data về business transaction được lưu trữ trong dimensional data warehouse để có thể make tactical and strategic business decisions

- Những người làm công việc này sẽ BA, business manager và executive

- Các chức năng sẵn có trong OLAP bao gồm aggregating (totaling), drilling down (getting the details) và slicing and dicing (cutting the cube and summing the values in the cells)

- Chức năng của OLAP có thể được luân chuyển bằng cách sử dụng relational database hoặc multidimensional database => Nếu dùng relational database thì gọi là ROLAP và với multidimensional database thì gọi là MOLAP

#### Data mining

- Data mining là quá trình khám phá dữ liệu để tìm ra những khuôn mẫu và mối quan hệ có thể mô tả chúng và dự đoán những giá trị chưa biết hoặc mang tính tương lai của data

- Key value của data mining là khả năng có thể hiểu được vì sao những điều này lại xảy ra trong quá khứ và dữ liệu những gì sẽ diễn ra trong tương lai => Nếu được dùng để giải thích các tình huống hiện có và quá khứ thì gọi là descriptive analytics, nếu đuợc dùng để dự đoán về tương lai thì gọi là predictive analytics

- Những ứng dụng của data mining trong BI là ngành tín dụng, tài chính, viễn thông, bán lẻ, bảo hiểm, năng lượng và chế tạo

### Other analytical activities

- Một số ứng dụng vào các hoạt động phân tích trong các mục đích phi kinh doanh là nghiên cứu khoa học, các cơ quan chính phủ (văn phòng thống kê, văn phòng thời tiết, phân tích kinh tế và dự đoán), tình báo quân sự, quản lý khẩn cấp và thảm họa, tổ chức từ thiện, giám sát hiệu suất máy chủ và phân tích lưu lượng mạng

- Data warehouse được sử dụng cho customer relationship management (CRM) => CRM là một set các hoạt động thực hiện bởi một tổ chức để quản lý và tiến hành các phân tíhc liên quan đến khách hàng của họ để có thể giữ liên lạc và kết nối tới họ, để thu hút khách hàng mới, tạo ra thị trường sản phẩm và dịch vụ cho khác hàng, tiến hành giao dịch, phục vụ và hỗ trợ khách hàng…

- Data warehouse còn được sử dụng trong web analytics => Web analytics là hoạt động có thể hiểu hành vi và đặc điểm của việc tắc nghẽn trên web site => bao gồm tìm hiểu số lượng truy cập, số người truy cập, các khách hành đặc biệt ở mỗi page cho mỗi ngày/tuần/tháng => Web analytics đặc biệt quan trọng cho việc kinh doanh online

### Update in batches

- Một data warehouse thường là một hệ thống chỉ cho phép đọc và cập nhật bằng việc sử dụng một cơ chế tiêu chuẩn là ETL ở những thời điểm cụ thể bằng cách đem data từ các hệ thống nguồn vận hành

- Lý do cho việc không cho phép user update và delete data trong data warehouse đó là có thể duy trì sự nhất quán của data nên sẽ đảm bảo rằng data trong data warehouse sẽ được đồng nhất với hệ thống nguồn vận hành

- Lý do thứ nhất mà data warehouse được update theo đợt thay vì theo thời gian thực đó là tạo ra sự ổn định của data trong data warehouse

- Lý do thứ hai đó là liên quan đến hiệu năng của hệ thống nguồn => nếu cập nhật theo thời gian thực thì sẽ phải đảm nhận khối lượng khổng lồ việc trích xuất dữ liệu và gần như không khả thi

=> Có một hướng tiếp cận đó là cập nhật real time nhưng chỉ với một vài bảng key

- Trong một vài năm trở lại đây, real-time data warehousing trở thành trend và thậm chí là tiêu chuẩn => Ngày trước là 1 ngày bây giờ chạy mỗi tiếng và một số cứ mỗi 5 phút (gọi là mini-batch)

=> Có hướng tiếp cận là push approach và pull approach => trong push approach thì DW update ngay lập tức khi data trong hệ thống nguồn thay đổi, còn trong pull approach thì DW được update tại khoảng thời gian cụ thế

- Một số hướng tiếp cận sử dụng messaging và công nghệ hàng đợi massage để vận chuyển data một cách có động bộ từ nhiều hệ thống nguồn tới DW

### Other difinitions

• According to Bill Inmon, a data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management’s decisions.

• According to Ralph Kimball, a data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making.

=> Với phương pháp của Inmon thì DW được cài đặt trong data store chuẩn hoá còn phương pháp của Kimball thì DW được cài đặt trong data store về chiều

- DDS nên được dùng để lưu trữ data trong DW nhằm mục đích truy vấn và phân tích data

- NDS nên được dùng để tích hợp data từ nhiều hệ thống nguồn

- Ngày nay, với sự xuất hiện của real-time data warehousing thì DW còn được dùng cho mục đích vận hành, ngoài việc make decision thì còn có hiểu các trường hợp cụ thể, cho mục đích báo cáo, tích hợp data và cho việc vận hành CRM

## Data warehousing today

### Business intelligence

- Ngày nay các nhà buôn bán ưu tiên các điều khoản về BI hơn là về DW, họ tập trung hơn vào việc DW có thể làm được những gì cho một doanh nghiệp

- Hầu hết DW được sử dụng cho BI, mục đích của DW là giúp business user có thể hiểu được doanh nghiệp của họ và đưa ra các quyết định vận hành và mang tính chiến lược, chiến thuật cho doanh nghiệp, đồng thời cải thiện hiệu quả kinh doanh

- BI system đã được xây dựng để giúp các quá trình này, bên cạnh đó là hiểu các khuôn mẫu về business data và dự đoán hành vi cho tương lai sử dụng data mining. Data mining cho phép doanh nghiệp tìm ra các khuôn dạng cụ thể trong dữ liệu và dự đoán các giá trị tương lai của data

### Customer relationship management (CRM)

- Một hệ thống CRM bao gồm các ứng dụng hỗ trợ cho hoạt động CRM. Trong hệ thống CRM những chức năng sau đây đã được hoàn thiện ở trong DDS:

* Single customer view
* Permission management
* Campaign segmentation
* Customer services/support
* Customer analysis
* Peronalization
* Customer loyalty scheme

- Cách chức năng khác như customer support hay order-processing support được phục vụ tốt hơn bởi operational data store (ODS) hay OLTP application

=> ODS là relational, normalized data store chứa transaction data và các giá trị hiện tại của master data từ OLTP system. Khi mà giá trị của master data trong OLTP system thay đổi thì ODS sẽ được cập nhật tương ứng sau đó. Một ODS tích hợp data từ một vài OLTP systems và không giống DW thì ODS được cập nhật thường xuyên hơn

### Data mining

- Data mining là lĩnh vực phát triển nhanh trong những năm gần đây => known as knownledge discovery, bởi nó bao gồm việc cố gắng tìm ra những thông tin có ý nghĩa và hữu ích từ một lượng data rất lớn. Nó là một quá trình tương tác và tự động để tìm các khuông mẫu mô tả data và dự đoán hành vi tương lai của data dựa vào những khuôn dạng đó

- Hệ thống data mining có thể làm việc với nhiều định dạng data khác nhau, từ database, files, unstructured hay semistructured datam, stream data, multimedia files…

- Ứng dụng data mining làm việc tốt nhất với DW bởi data đã được clean, có cấu trúc và có metadata mô tả data, được tích hợp và không bay hơi (dữ liệu tĩnh), và điều quan trọng nhất đó là chúng được sắp xếp trong dimensional format phù hợp với rất nhiều data mining tasks

- Trong các dự án data mining, data từ nhiều nguồn đã được đề cập ở trên sẽ được xếp vào trong một dimensional database, ứng dụng lấy data từ database này để ứng dụng các thuật toán data mining và logic và data, sau đó hiển thị kết quả tới end users

- Các ứng dụng business và nonbusiness của data mining:

* Finding out which products are likely to be purchased together
* In the railway or telecommunications area, predicting which tracks or networks of cables and switches are likely to have problems this year
* Finding out the pattern between crime and location and between crime rate and various factors, in an effort to reduce crime
* Customer scoring in CRM in terms of loyalty and purchase power
* Credit scoring in the credit card industry
* Investigating the relationship between types of customers and the services/products they would likely subscribe to/purchase
* Creating a call pattern in the telecommunication industry

- Để cài đặt data mining trong SQL Server Analysis Services (SSAS), bạn sẽ build một mining model sử dụng data từ relational sources hoặc từ OLAP cubes chứa thuật toán mining cụ thể như cây quyết định hay clustering

### Master Data Management (MDM)

- Để hiểu được MDM là gì thì chúng ta cần biết master data là gì. Trong hệ thống OLTP, có 2 dạng data là transaction data và master data

- Transaction data bao gồm các thực thể của doanh nghiệp trong hệ thống OLTP mà ghi lại các business transaction gồm việc định dạng, giá trị và các cột thuộc tính

- Master data bao gồm các thực thể của doanh nghiệp ở trong hệ thống OLTP mà mô tả các business transaction bao gồm việc định dạng và các cột thuộc tính

=> Transaction data được liên kết với master data nên master data mô tả business transaction

- Để hiểu được thực thể nào là transaction data hay master data thì cần phải mô hình quy trình kinh doanh => business event là transaction data. Master data bao gồm các thực thể mà mô tả business event => Master data bao gồm các câu trả lời về who, what và where về business transaction

🡺 Có thể thấy để xác định transaction data và master data trong một quy trình kinh doanh thì bạn phải xác định được trước tiên đâu là business event trong quy trình, sau đó sẽ xác định các thực thể business mô tả business event đó

- Examples of master data are the supplier, branch, office, employee, citizen, taxpayer, assets, inventory, store, salespeople, property, equipment, time, product, tools, roads, customer, server, switch, account, service code, destination, contract, plants (as in manufacturing or oil refineries), machines, vehicles, and so on

=>> MDM là quá trình liên tục của việc truy xuất, làm sạch, lưu trữ, cập nhật và phân phối dữ liệu chính. Hệ thống MDM truy xuất dữ liệu chính từ hệ thống OLTP. Hệ thống MDM hợp nhất dữ liệu chủ và xử lý dữ liệu thông qua các quy tắc chất lượng dữ liệu được xác định trước. Dữ liệu chính sau đó được tải lên kho dữ liệu chính. Bất kỳ thay đổi nào về dữ liệu chính trong hệ thống OLTP đều được gửi đến hệ thống MDM và kho lưu trữ dữ liệu chính được cập nhật để phản ánh những thay đổi đó. Sau đó, hệ thống MDM xuất bản dữ liệu chính cho các hệ thống khác

- Có 2 dạng master data mà bạn không cần phải thêm trong quá trình cài đặt một hệ thống MDM là:

1, data and time

2, master data with a small number of members

- Master data được xác định vị trí ở trong hệ thống OLTP, những master data này liên tục thay đổi, thay đổi flow từ hệ thống OLTP đến master data store trong hệ thống MDM. Có 2 con đường khả thi để data flow này diễn ra là push approach và pull approach:

* Push approach là nơi hệ thống OLTP gửi sự thay đổi của master data tới hệ thống MDM
* Pull approach là nơi hệ thống MDM lấy master data từ hệ thống OLTP theo định kì

- Hệ thống MDM có metadata storage. Metadata storage là database chứa các quy định, cấu trúc và ý nghĩa của data => Mục đích là giúp user hiểu được ý nghĩa và cấu trúc của master data được lưu trữ trong hệ thống MDM. Có 2 dạng rule được lưu trữ trong metadata storage là survivorship rules và matching rules:

* Survivorship rules xác định master data record nào trong 2 bản duplicate trong từ hệ thống OLTP sẽ được giữ lại trong hệ thống MDM như một master data
* Matching rules xác định những thuộc tính gì được sử dụng để định dạng duplicate record từ hệ thống OLTP

- Cấu trúc dữ liệu lưu trữ trong metadata storage giải thích các thuộc tính của master data và data types của từng thuộc tính

- Hệ thống MDM có một công cụ báo cáo hiện các cấu trúc dữ liệu và, survivorship rules, matching rules và duplicate records từ hệ thống OLTP cùng những rule đã được ứng dụng và record được giữ lại là master data => reporting facility cũng hiển thị những rule được khai thác và khi nào được khai thác

- Hệ thống MDM quản lý dữ liệu sản phẩm được gọi là product information management (PIM) => PIM là hệ thống MDM lấy các data sản phẩm từ hệ thống OLTP, dọn dẹp và lưu trữ data sản phẩm ở trong master data store => duy trì toàn bộ các thuộc tính sản phẩm và sự phân cấp sản phẩm ở trong master data store giữ chúng luôn được cập nhật ở bằng cách lấy các thay đổi từ hệ thống OLTP và publish data sản phẩm đến các hệ thống khác

- Vậy MDM có liên quan gì đến với DW? Sự liên quan đến từ 2 điều sau:

1, Nếu toàn bộ master data là clean thì nó sẽ giúp việc tạo và duy trì một DW được dễ dàng hơn

2, Bạn có thể sử dụng DW để lưu trữ master data

- Có một điều cần phải thảo luận nữa, đó là sự phân cấp. Hierachy là cấu trúc mà ở đó master data được chia thành các nhóm danh mục, và các danh mục này được phân loại thành level liên quan. Top-level category có 1 hoặc nhiều second-level categories và mỗi second-level category lại có 1 hoặc nhiều second-level categories, cứ tương tự như vậy. Đến lowest-level category có master data cá nhân => Hierachy được sử dụng để xác định thuộc tính nào được ứng dụng vào data nào và khi di chuyển một item từ category này đến category khác

### Customer data intergration

- Customer data intergration (CDI) là MDM cho data khách hàng

- CDI là quá trình lấy, dọn dẹp, lưu trữ, duy trì và phân phát data khách hàng. Hệ thống CDI lấy data khách hàng từ hệ thống OLTP, clean và store chúng ở trong customer master data store, duy trì data khách hàng, giữ nó được cập nhật và phân phát data khách hàng tới hệ thống khác

- Một hệ thống CDI cho phép bạn có được một phiên bản đáng tin, độc lập, và gọn gàng của data khách hàng mà ứng dụng khác trong doanh nghiệp có thể sử dụng

- CDI được sử dụng nhiều nhất và rộng rãi nhất bởi tất cả các tổ chức đều có tệp khách hàng riêng, cung cấp data đã được tích hợp và dọn dẹp cho CRM

## Future trends in data warehousing

### Unstructured data

- Unstructured data không có cấu trúc dữ liệu như cột hay dòng, cấu trúc cây, hay class và types

- Ví dụ về unstructured data là documents, images, audio, video, streaming data, text, emails và internet web site

- Một số người có thể coi những data này là semistructured data bởi chúng vẫn có một số cấu trúc và sẽ có các thuộc tính

- Vậy làm thế nào lưu trữ unstructed data ở trong DW? Sau đó làm thế nào để lấy nó ra khi cần?

=> Câu trả lời sẽ là định nghĩa các thuộc tính của unstructured data và rồi tổ chức các item theo các thuộc tính đó. Bạn có thể lưu trữ các unstructured data item ở trong relational database như một binary object column và các thuộc tính là các column khác => Ngoài ra còn có thể lưu trữ trong các hệ thống file và chỉ cần lưu trữ con trỏ trỏ tới file ở trong database

- Unstructured data dù sao cũng không phải là một điều gì đó quá mới, bởi phân tích văn bản có từ rất lâu. Và mới đây con người mới nhận ra rằng là hầu hết dữ liệu tồn tại ở dưới dạng unstructured data, đặc biệt là với văn bản như document và email

### Search

- Đây là câu trả lời cho câu hỏi bạn lấy các thông tin ra như thế nào => by searching

- Để có thể lấy được thông tin về structured data, cần chắc rằng bạn hiểu về cấu trúc dữ liệu để có thể lựa chọn truy vấn hay sử dụng một báo cáo cố định hoặc truy vấn tương tác thủ công => Nếu sử dụng một ứng dụng BI => ứng dụng có thể đi đến metadataa và hiển thị cấu trúc của dữ liệu và rồi hỗ trợ bạn điều hướng tới data để lấy thông tin bạn cần

- Đối với unstructured data, các công cụ search đã được tích hợp dữ liệu ở DW và đánh dấu chỉ mục các unstructured data => phân loại chúng dựa vào types và properties và trong trường hợp với webpage thì dựa vào links

- Search ở trong thế giới internet đã có mặt được khoảng 10-15 năm nhưng ở trong data warehouse mới chỉ mới thôi 2-3 năm

- Khi mọi người collect các unstructured data với lượng lớn và việc search rất dễ dàng để thực hiện kể cả với structured data => search hiện nay trở thành một trend ở trong BI và DW

### Service-oriented architecture (SOA)

- SOA là một phương pháp xây dựng một ứng dụng sử dụng một số lượng các thành phần nhỏ và độc lập có thể tương tác lẫn nhau bằng cách đề xuất và sử dụng các dịch vụ của chúng

- Các thành phần này được phân phát, trong thực tế chúng nằm ở nhiều phần khác nhau

- Hầu hết các ứng dụng lớn có thể đặt được sự hữu dụng từ phương pháp SOA khi không nhất thất phải build một ứng dụng khổng lồ. Thay vào đó, bạn xây dựng các phần nhỏ có thể tương tác với nhau => Nó rất phù hợp với ngành IT hiện nay khi sự thay đổi diễn ra và các công nghiệp sẽ thay thế các công nghệ cũ sau một vài năm

- Nếu tạo ra một ứng dụng khổng lồ thì chi phí để thay thế chúng là rất lớn và cũng không dễ để thay thế. Với SOA, chúng ta sẽ thay thế các thành phần dễ dàng nên có sự linh hoạt, các thành phần khi thay thế sẽ không ảnh hưởng đến quá trình vận hành của những phần còn lại vì chúng hoạt động độc lập và không ảnh hưởng đến nhau

- Vậy SOA có liên quan gì đến DW? Bạn có thể thấy rằng một hệ thống DW có rất nhiều thành phần. Nếu bạn build chúng như một ứng dụng khổng lồ thì việc thay thế một thành phần mà không ảnh hưởng đến những thành phần khác là không thể. Ngược lại, nếu bạn build chúng theo kiến trúc hướng dịch vụ thì khi các thành phần dù có bị thay thế thì các bộ phận khác vẫn vận hành bình thường

- Ngày càng nhiều ứng dụng DW đang sử dụng SOA: ETL, reporting, analytics, BI applications, data mining, metadata, data quality và data cleansing => Trong tương lai, sẽ dễ dàng để update một thành phần mà không có ảnh hưởng đến những phần còn lại và có thể kết nối lại mặc cho sử dụng các công nghệ khác nhau

### Real-time data warehouse

- Một DW ở vài năm trước thường được updated theo ngày hoặc theo tuần nhưng trong 2-3 năm trở lại đây đã có sự ra tăng về nhu cầu muốn tăng tuần suất update => Users muốn được thấy data trong DW được update mỗi 2 phút và thậm chí theo thời gian thực => Real-time data warehouse là DW được update bởi ETL ở mỗi lúc có một transaction diễn ra trong hệ thống nguồn

=> Ví dụ, có thể đặt triggers ở sales transaction table

- Một hướng tiếp cận khác đó là modify operational source application để viết vào DW staging area, ngay lập tức sau khi chúng viết data ở trong internal database

- Near real-time approach có thể được cài đặt bằng cách sử dụng mini-batch với tần suất 2-5 phút, khi kéo data từ stage area thay vì sử dụng triggers => mini-batch cũng sử dụng normal ETL job là biến đổi data và tải chúng lên DW dimensional database. Mini-batch cũng kéo data trực tiếp từ hệ thống nguồn, loại bỏ việc cần biến đổi hệ thống nguồn để update stage area

## Summary

- Chương này đã giới thiệu về data warehouse, với các ví dụ để giúp cho việc hiểu và làm phong phú hơn về trải nhiệm

- Các giải pháp cho hiện tại, các xu hướng của tương lai cũng đã được đề cập

- Ở chương tiếp theo chúng ta sẽ tìm hiểu về architecture

# Data Warehouse architecture

## Data flow architecture

- Data flow architecture is configuration of data stores within a Data warehouse system, along with arrangement of how the data flows from the source sytems through these data stores to the application used by the end users

=> Data flow architecture is different from data architecture

=> Data architecture is about how the data is arranged in each data store and how a data store is designed to reflect the business processes => activity to produce data architecture is known as data modeling

- Data store là một thành phần quan trọng của Data flow architecture => Là một hay nhiều csdl hay files chứa dữ liệu data warehouse, được sắp xếp theo một format cụ thể và nằm trong tiến trình data warehouse

=> Dựa vào khả năng truy cập thì sẽ phân loại data warehouse data store thành 3 loại

* *User-facing data store* => cho phép end user được truy cập và tiến hành truy vấn cùng với end-user applications
* *Internal data store* => được sử dụng nội bội bởi các thành phần data warehouse cho mục đích intergrating, cleansing, logging and preparing data => not open for query by end user and end user applications
* *Hybird data store* => both

- Master data store là user-facing or hybrid data store chứa complete set of data trong data warehouse, bao gồm all version and all historical data

=> Dựa vào data format thì phân loại data warehouse data store thành 4 loại:

* *Stage* => internal data store => used for transforming and preparing the data obtained from source systems, before loading to other data store in data warehouse
* *Normarlized data store* => internal master data store => form của 1 hay nhiều normalized relational database cho mục đích tích hợp data từ nhiều source system được lưu trong một stage, before loading data to user-facing data store
* *Opretional data store* => hybrid data store => form của 1 hay nhiều normalized relational database chứa transaction data và hầu hết các version hiện tại của master data, cho mục đích hỗ trợ ứng dụng vận hành
* *Dimensional data store* => user-facing data store => form của 1 hay nhiều relational database, nơi data được sắp xếp theo dimensional format cho mục đích hỗ trợ các truy vấn về phân tích

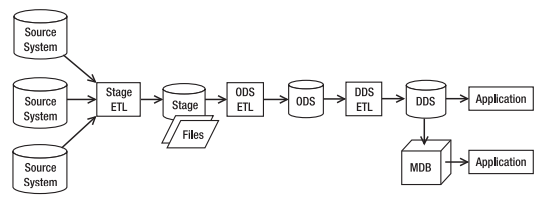
- Chúng ta cũng sẽ nói lại về các dạng database:

• A *relational database* is a database that consists of entity tables with parent-child relationships between them.

• A *normalized database* is a database with little or no data redundancy in third normal form or higher.

• A *denormalized database* is a database with some data redundancy that has not gone through a normalization process.

• A *dimensional database* is a denormalized database consisting of fact tables and common dimension tables containing the measurements of business events, categorized by their dimensions



- Phía trên là một Data flow architecture có cả 4 dạng data store

=> Quy trình sẽ như sau:

1, Stage ETL lấy data từ source system rồi load lên stage

2, Opretional data store ETL lấy data từ stage rồi load lên Opretional data store

3, Dimensional data store ETL lấy data từ Opretional data store rồi load lên Dimensional data store

- Một ETL package sẽ bao gồm một vài ETL processes => là 1 phần của ETL package đảm nhận việc lấy data từ một vài nguồn và load chúng lên một target table

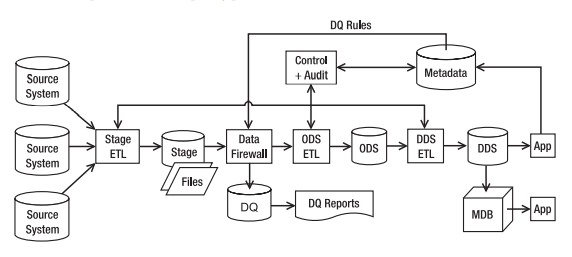
- Một ETL processes gồm một vài ETL steps => mỗi step lại đảm nhận một task riêng

- Control system quản lý ETL package

- ETL audit là cơ chế để đánh giá kết quả từng bước một ETL process

- Mô tả của từng ETL process được lưu trữ trong metadata => including source it extracts the data from, the target it loads the data into, the transformation applied, the parent process, and the schedule each ETL process is supposed to run

- Trong data warehousing, metadata là một data store chứa các mô tả về structure, data, processes ở trong data warehouse, bao gồm định nghĩa dữ liệu và ánh xạ, data structure của từng data store, source system, mô tả của từng ETL process, data quality rules, đăng nhập của toàn bộ quá trình và hoạt động trong data warehouse



- Phía trên là mô hình Data flow architecture khi có control system, metadata và data quality process

=>> Data flow architecture là một trong những điều đần tiên bạn cần phải quyết định khi xây dựng một Data warehouse system vì Data flow architecture xác định những thành phần cần được xây dựng và sẽ ảnh hưởng để kế hoạch dự án và chi phí. Data flow architecture cho biết được data flow luân chuyển qua các data store ở trong Data Warehouse

- Chúng ta có 4 kiến trúc chủ yếu của Data flow architecture đó là:

• The *single DDS architecture* has stage and DDS data stores.

• The *NDS + DDS architecture has stage*, NDS, and DDS data stores.

• The *ODS + DDS architecture has stage*, ODS, and DDS data stores.

• The *federated data warehouse (FDW) architecture* consists of several data warehouses integrated by a data retrieval layer

### Single Dimensional data store

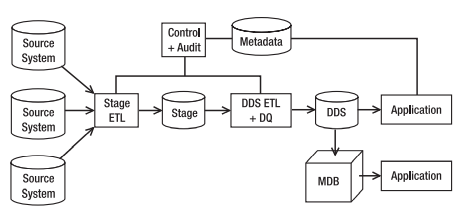
- Đây là Data flow architecture đơn giản nhất khi chỉ có 2 data store là stage và Dimensional data store và lõi của data warehouse store trong kiến thúc này là dimensional format

- Với single DDS architecture thì DDS gồm một vài dimensional data mart

- Một dimensinal data mart là một nhóm các bảng fact và các bảng dim tương ứng chứa những dữ kiện về business event, được phân loại bởi các chiều của chúng

- Stage là nơi bạn lưu trữ tạm thời data được trích xuất từ source system, trước khi xử lý chúng ở những công đoạn sau => cần thiết khi mà tranformation phức tạp, khi data volume là lớn hoặc khi data từ các source system đến ở những thời gian khác nhau => cần thiết khi muốn giảm đáng kể thời gian trích xuất data từ source system

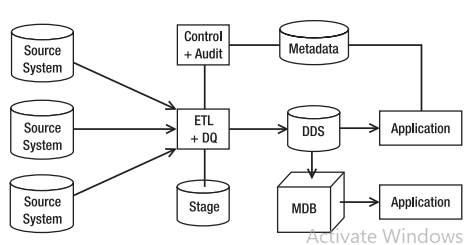
- Dạng vật lý của stage có thể là database hoặc là files => ETL trích xuất data từ source system thêm data vào một database hoặc viết nó như là files => ETL thứ 2 sẽ lấy data từ stage, tích hợp những nguồn khác, áp dụng các rules và đẩy các data được tập hơn lên DDS



- Phía trên là mô hình của Single DDS data warehouse architecture, trong đó:

* “Control + Audit” box chứa ETL control system và audit => quản lý ETL processes và log ETL execution results
* Metadata database chứa description về structure, data và processes trong Data Warehouse

- Bên cạnh đó, chúng ta có thể combine stage ETL, DDS ETL và data quality processes thành 1 => ưu điểm là có thể kiểm soát thời gian data được viết và lấy từ stage => nhược điểm là làm cho ETL package trở nên phức tạp hơn



🡺 Ưu điểm của single DDS architecture đó là nó đơn giản hơn các kiến trúc còn lại, khi data được load thằng từ stage lên DDS

🡺 Nhược điểm đó là rất khó để có thể tạo ra scecond DDS => Nếu muốn tạo một DDS nhỏ hơn thì sẽ phải tạo mới ETL package chứ không thể tái sử dụng ETL package đang có

- Trong kiến trúc này, DDS là master data store

🡺 Bạn sẽ nên dùng single DDS architecture khi bạn chỉ cần một dimensional store và không cần NDS và khi bạn chỉ có 1 source system

### NDS + DDS

- Trong kiến trúc này chúng ta có 3 data store đó là stage, NDS và DDS

- Mô hình sẽ tương tự như single DDS nhưng sẽ có thêm NDS ở giữa stage và DDS

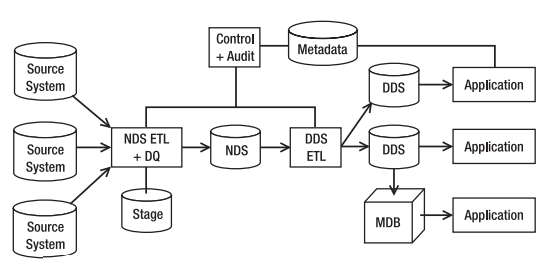
- NDS này ở third form hoặc cao hơn

- Việc có NDS trong kiến trúc này có 2 mục đích:

1, Tích hợp data từ nhiều source system

2, Cho phép load data lên nhiều DDS

=>> Điểm khác biệt ở đây là NDS + DDS architecture có thể có nhiều DDS trong mô hình như ví dụ dưới đây:



- Trong kiến trúc này, NDS là master data store

- Giống như OLTP source system, có 2 dạng table trong NDS là transaction table và master table

=> NDS transaction tables là DDS fact table và NDS master table là NDS dimension table

- Các nguyên tắc về stage, control, audit, metadata đều được áp dụng

- Một vài entities có thể lên NDS mà không cần phải đẩy lên stage trước đó. Trong trường hợp này thì việc tích hợp và chuyển đổi data được thực online trong memory của ETL server

- Sự linh hoạt khi sử dụng một NDS trung tâm đó là có thể xây dựng một NDS bất kì khi nào với những yêu cầu về độ rộng của data => hữu ích để làm hài lòng các yêu cầu từ các dự án cần có sự phân tích về data

*- The ability to set the scope of data when building a new DDS means you can pick which tables, columns, and rows you want to transfer to the new DDS*

- Việc build một DDS thì có thể dùng DDS ETL hiện có, chỉ cần trỏ ETL đến DDS mới. Nếu bạn build DDS ETL đúng thì bạn có thể xây dựng lại một DDS bất kì rất nhanh chóng

=>> Để có được sự linh hoạt này thì DDS ETL cần được tham số hoá

- Trong kiến trúc này thì bạn có thể có nhiều DDS nhưng sẽ phải có một DDS với một số bổn phận được xây dựng và duy trì, đây là DDS chứa toàn bộ bảng fact và bảng dim

🡺 Ưu điểm chính của kiến trúc này là:

1, có thể dễ dàng xây dựng lại main DDS, ngoài ra là có thể build mới một DDS khác, khi mà NDS mới là master data store và DDS ETL được tham số hoá

2, Dễ dàng để maintain master data trong normalized store như NDS và publish từ đây, bởi chúng có ít hoặc thậm chí không có data redundancy và bạn chỉ cần update một nơi trong data store

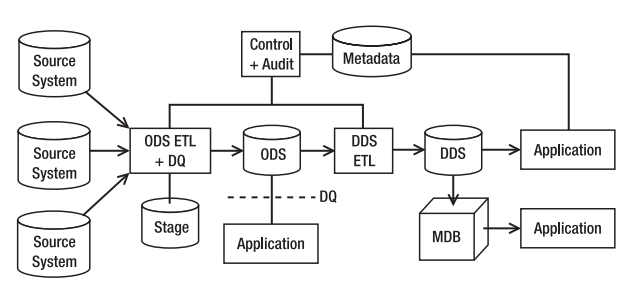
🡺 Nhược điểm chính của kiến trúc đó là yêu cầu nhiều nỗ lực để xây dựng khi mà data từ stage cần được đẩy lên NDS trước khi được upload lên DDS

🡺 Bạn sẽ sử dụng NDS + DDS architecture khi bạn cần có các DDS khác nhau cho mục đích khác nhau và cần tích data trong một form chuẩn hoá và sử dụng chúng bên ngoài dimensional data warehouse

### ODS + DDS architecture

- Kiến trúc này giống với NDS + DDS architecture, với ODS thay thế cho vị trí của NDS. ODS ở third form hoặc cao hơn

- Khác với NDS thì ODS chỉ chứa version hiện tại của master data mà không có historical master data



- Phía trên là mô hình của kiến trúc ODS + DDS, tương tự với NDS thì ODS gồm có transaction tables và master tables, trong đó fact tables của DDS được tạo ra từ transaction tables của ODS, dimension tables của DDS được tạo ra từ master table của ODS

- ODS là hybrid data store => cho phép end users và end-user applications có thể truy cập vào để lấy data từ ODS. Đặc biệt, vì ODS luôn được cập nhật nên end-users có thể cập nhật ODS, nhưng không được cập nhật data đến từ các source system mà chỉ được cập nhật data mà sinh ra từ chính nó sau này để bổ nghĩa cho data của source system

- DDS sẽ là master data store của kiến trúc này. Giống với kiến trúc single DDS thì ODS + DDS architecture chỉ có một DDS chứa toàn bộ các bảng fact và dim và chứa toàn bộ các version của master data

- Các nguyên tắc đều được thực hiện giống như 2 kiến trúc trên, và một số thực thể cũng có thể được đem trực tiếp đến ODS mà không cần ở stage trước (giống như NDS + DDS architecture)

- DDS ETL sẽ đơn giản hơn ETL package ở trong single DDS bởi data đã được tích hợp và cleaned

- Trong kiến trúc này, applications có thể truy cập và Data Warehouse ở 3 nơi theo 3 format khác nhau đó là ODS, DDS và MDBs tương ứng với dạng data mà chúng cần

🡺 Ưu điểm của kiến trúc này:

* Third normal form sẽ gọn gàng hơn NDS khi nó chỉ chứa gia trị hiện tại và nó làm cho hiệu suất của cả ODS ETL và DDS ETL ổn hơn ETL tương ứng của NDS + DDS architecture
* Có một central place để tích hợp, duy trì ổn định và publish master data
* Normalized relational store được update bởi end-user application => making it capable of supporting operational applications at the transaction level

🡺 Nhược điểm của kiến trúc này đó là nếu muốn build một DDS mới và nhỏ thì sẽ phải lấy từ từ DDS chính chứ không thể tận dụng DDS ETL đang tồn tại (khá tương đồng với single DDS architecture)

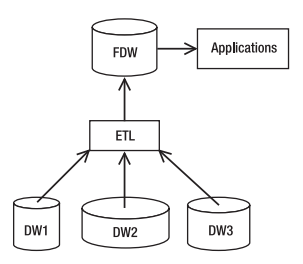
🡺 Bạn sẽ sử dụng kiến trúc này khi bạn chỉ cần một DDS và bạn cần một centralized, NDS được sử dụng cho mục đích vận hành như CRM. ODS chứa các data chi tiết, chứa giá trị hiện tại và đã được tích hợp thì sẽ hữu ích cho transactional queries

### Federated data warehouse

- FDW bao gồm một vài Data Warehouse với một data retrieval layer on top of them => FDW lấy data từ các Data Warehouse hiện có sử dụng một ETL và tải data lên một DDS mới

- Độ chi tiết của FDW data là tương đương với the highest of granularities of source data warehouse

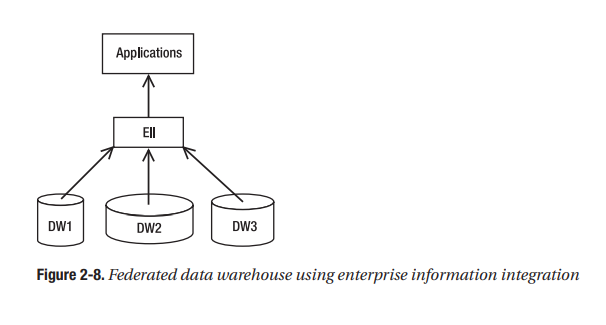
- ETL mà trích xuất data từ source data warehouse cần có nhận thức về mặt thời gian của data => Data trong source data warehouse có thể không đến với tần suất giống nhau nên FDW ETL cần phải match cái tần suất này với scource Data Warehouses, nếu không sẽ phải chạy vài lần mới có thể match được



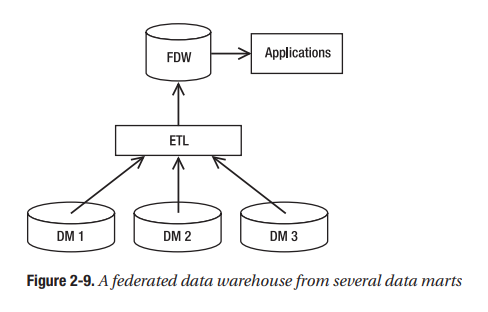
- FDW ETL cần tích hợp data từ source Data Warehouses dựa theo business rules. ETL cần xác định rõ record nào từ SDW này trùng với record từ các SDW khác => Duplicate record cần được merge lại và business rules cần được áp dụng để xác định record được giữ lại và thuộc tính nào cần được update. Ngoài ra thì FDW ETL cần phải transform data từ nhiều SDWs thành các structure và format giống nhau

- Khi mà có một vài SDW thì hoàn toàn có thể cài đặt enterprise information integration (EII) => EII là phương thức tích hợp data bằng cách truy cập trực tuyến nhiều source system khác nhau và tổng hợp outputs on the fly trước khi đem kết quả cuối đến users => Mọi thứ đều được hoàn thành vào thời điểm mà user vừa yêu cầu về thông tin. Bên cạnh đó chúng ta không cần một storage hay form nào để lưu trữ data đã được tổng hợp và tích hợp

🡺 Ưu điểm chính của việc dùng EII là data freshness



- Ngoài ra, thay vì tích hợp một vài Data Warehouse thì FDW có thể được cài đặt khi mà có một vài nointegrated data mart ở trong tổ chức



🡺 Ưu điểm chính của kiến trúc này đó là cung cấp Data Warehouse hiện có và thời gian phát triển sẽ rút ngắn hơn

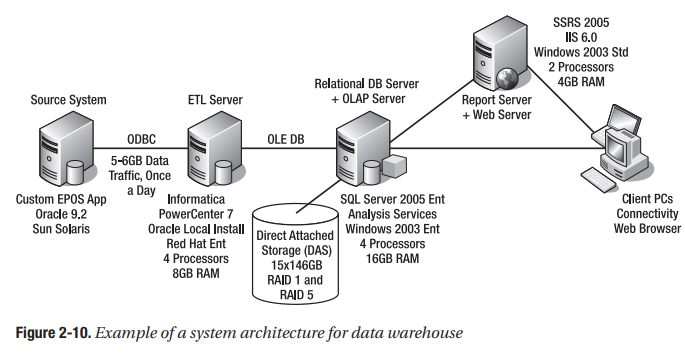
🡺 Nhược điểm chính đó là khó mà có thể xây dựng được một warehouse có chất lượng tốt từ những tiêu chuẩn đa dạng được tìm thấy từ source data marts hay Data Warehouse

🡺 Bạn sẽ sử dụng FDW khi bạn muốn tận dụng những Data Warehouse hiện có và ở những nơi bạn muốn tích hợp data từ nhiều data mart

## System architecture

- Một khi mà đã chọn được một Data flow architecture cụ thể thì bạn sẽ cần thiết kết system architect => đó là sự sắp xếp về mặt vật lý và kết nối giữa servers, network, software, storage system và clients

- Thiết kế một system architecture yêu kiến thức về phần cứng (đặc biệt là về server), networking (fiber network) và storage



- Để thiết kế một Data Warehouse system architecture:

* Cần xây dựng những thành phần công nghệ bạn muốn dùng cho ETL, database và BI => phụ thuốc vào product capability và tiêu chuẩn doanh nghiệp
* Sau khi đã xác định được technology stack => do a high-level desgin trên servers, network configuration, storage configuration hỗ trợ cho công nghệ đã được chọn, bao gồm high-availability design => Phụ thuộc vào capacity và system performance requirements
* Order hardware và software và build hệ thống trong data center => Install và configure software, lưu ý rằng việc chọn phần mềm cũng rất quan trọng và ảnh hưởng lớn để đến system architecture
* Designing và building environment là quan trọng và đặc biệt thiết yếu cho hiệu suất và sự ổn định của Data Warehouse system mà bạn đang build

- Với phần mềm => có 2 dạng database software là symmetric multi-processing (SMP) và massively paraller processing (MPP)

* SMP database system là database system mà chạy trên trên 1 hay nhiều máy với các tiến trình chia sẻ chung một bộ nhớ lưu trữ
* MPP database system là database system chạy trên nhiều máy và mỗi máy đều có bộ nhớ lưu trữ riêng

- Các máy ở trong 2 dạng trên được gọi là các node

🡺MPP database system nhanh hơn và có thể thay đổi kích thước tốt hơn SMP database system, nhưng SMP database system đơn giản hơn, dễ dàng duy trì và chi phí cũng rẻ hơn

## Case study

- Case study này sẽ được sử dụng xuyên suốt cuốn sách, bao hàm toàn bộ những kiến thức và khía cạnh chúng ta muốn học, về kiến trúc, phương pháp, yêu cầu, data modeling, thiết kế csdl, BI, CRM, kiểm thử, quản trị Data Warehouse

- Tất cả mọi thứ đều được tối ưu và lý tưởng cho case study này

- Amadeus Entertainment is an entertainment retailer specializing in music, films, and audio books. It has eight online stores operating in the United States, Germany, France, the United Kingdom, Spain, Australia, Japan, and India. It has 96 offline stores operating in those countries as well.

- Customers can buy individual products such as a song, an audio book, or a film, or they can subscribe to a certain package, which enables them to download a certain quantity of products in a certain period. For example, with the Primer package, you can download 100 songs, 50 books, and 50 films a month for $50. Customers can also listen or watch a song, a book, or a film once for 1 ⁄10 the cost of purchasing it. So if a film is $5, to watch it once it’s only 50 cents. Customers use online streaming for this, so a good Internet connection is required.

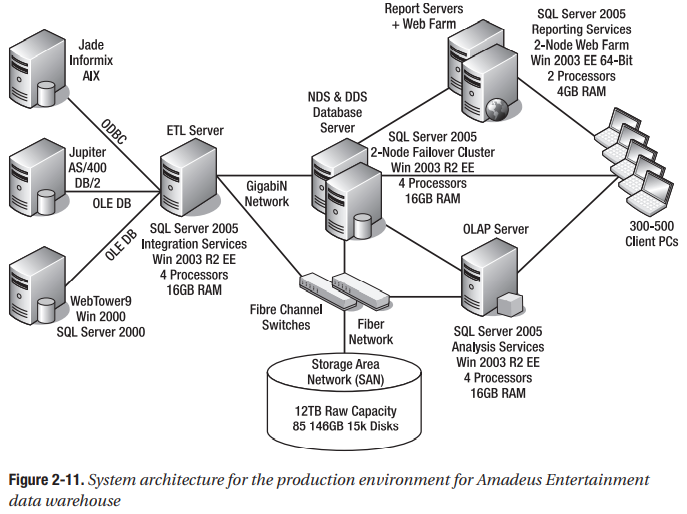
- Amadeus Entertainment has four main delivery channels: Internet, mobile phone, cable TV, and post. There are several payment models for customer subscriptions, such as annual, in advance, and monthly direct debit. The company purchases the products in bulk, such as any 10,000 songs from a record company, of any title, for a certain cost. For online streaming, the company pays a central provider (Geo Broadcasting Ltd.) based on usage (monthly invoice).

- The company uses WebTower9, a custom-developed .NET-based system for dynamic web sites, multimedia trading, broadcasting, sales order processing, and subscription management, all running on an Oracle database. The back-end enterprise resource planning (ERP) system is Jupiter, an off-the-shelf AS/400-based business system running on DB2. This is where the inventory, products, and finances are managed. The Jupiter database size is about 800GB with about 250 tables and views.

- Business activities in the offline stores are managed in Jade, which is a custom Java-based system running on Informix. This includes sales, customer service, and subscriptions. WebTower9 and Jade interface with Jupiter products and finances on a daily basis, but sales and customer data (including subscriptions) are kept on WebTower9 and Jade.

- The company uses the SupplyNet system for interfacing with suppliers. It is a web services–based industry-standard supplier network in the online entertainment industry including music and film. WebTower9 and Jupiter are hosted in Amadeus Entertainment’s head office. Jade is hosted in an outsourced company in Bangalore, India. The Jupiter overnight batch starts at 11 p.m. Eastern standard time (EST) for three to four hours. An offline backup job kicks off immediately after the overnight batch and runs for one to two hours. The Jade tape backup starts at 3 a.m. EST for two to three hours.

- The IT architecture team decided to standardize the database platform on Microsoft SQL Server and use web-based services for its application platform. Hence, in this case study, I will show how to build a data warehouse system for Amadeus Entertainment using Microsoft SQL Server 2005. You can also use SQL Server 2008 to build it. I will be using NDS + DDS data warehouse architecture outlined earlier in this chapter. Figure 2-11 shows the system architecture for the production environment



## Summary

- Ở chương này, chúng ta đã thảo luật về các kiến trúc trong Data Warehouse, bao gồm các data flow architecture là single DDS, NDS + DDS, ODS + DDS và federated Data Warehouse.

- Chúng ta đã thảo luận về system architecture

- Với case study trên, chúng ta sẽ dùng chúng để hiểu sâu hơn những kiến thức ở các chương sau

# Data Warehouse Development Methodology

- In software engineering, the discipline that studies the process people use to develop an information system is called the system development life cycle (SDLC) or the system development methodology.

- There are two main variants: waterfall and iterative. The waterfall methodology is also known as the sequential methodology. The iterative methodology is also known as the spiral or incremental methodology

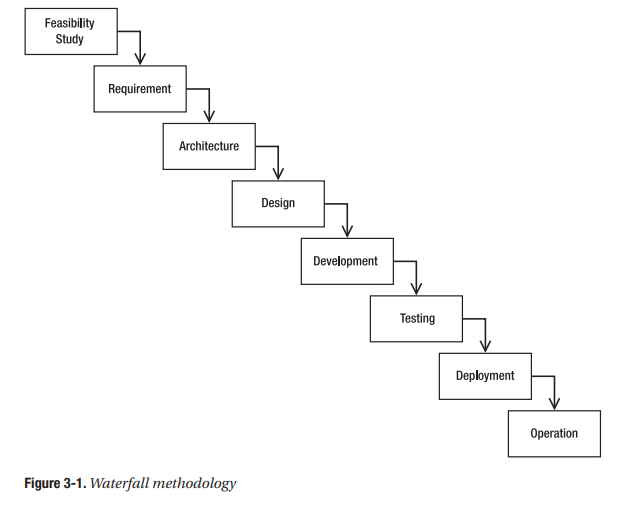
## Waterfall methodology

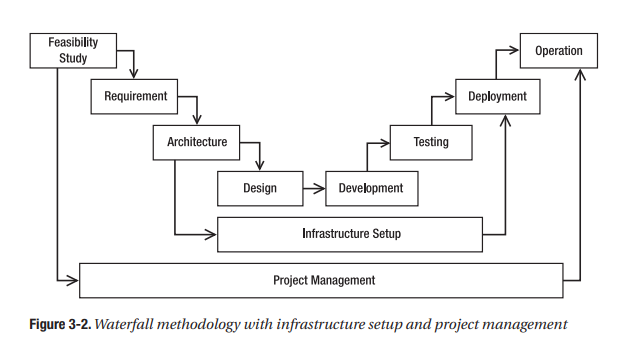
- Với hệ thống này, bạn có một các bước cụ thể cần phải hoàn thành trước khi đến với công đoạn khác. Các tên gọi chung của các bước đó là:

1. Feasibility study
2. Requirements
3. Architecture
4. Design
5. Development
6. Testing
7. Deployment
8. Operation

- Một số tên của các bước cũng có thể được đề cập ở trong hệ thống, bên cạnh đó có một số quá trình được chia thành nhiều phần nhỏ hơn => Requirements chia thành functional và non-functional requirements; Design chia thành functional, technical và architecture design; testing chia thành unit, system, integration, user acceptance testing

- Ngoài ra chúng ta có thể thêm vào 2 quá trính là “Infrastructure Setup” và “Project Management” => Khi đó hệ thống được thể hiện theo mô hình chữ V





### Feasibility study

- Bạn sẽ đi thu thập các yêu cầu ở mức độ cao (ví dụ: xác định xem vì sao mình cần Data Warehouse và việc xây dựng chúng có phải là giải pháp đúng?)

- Bạn một cái nhìn sơ qua vào source system để xem xem có khả thi hay không việc lấy được data mà bạn cần  
- Lấy một mẫu data để đánh giá data quality và sau đó viết đề xuất (proposal) a.k.a business case

=> Những điều quan trọng cần có trong đề xuất này đó là benefits, taken time, price, business justifications và có thể cả requirements, project organization và project plan

=> Để có thể có một đề xuất hoàn chỉnh cần có kiến thức trước về toàn bộ các mảng trong hệ thống, về thời gian cài đặt và cả chi phí bỏ ra => Người chịu tránh nhiệm cho công đoạn này phải có kinh nghiệm và đã xây dựng nhiều Data warehouse system

- Bởi vậy, việc coi bước này là một project riêng và có ngân sách riêng là một ý tưởng cực tốt => Nếu lựa chọn cách này thì bạn sẽ cần có một đội ngũ gồm một số BA phân tích yêu cầu, một Data Warehouse architect xác định kiến trúc, một project manager đảm nhiệm ước lượng và project plan ở high level, một hardware and infrastructure person, một business owner và source system DBA

### Requirements

- Bạn sẽ cần trao đổi với người dùng chi tiết về quy trình, về nghiệp vụ, về data và về các vấn đề có thể phát sinh, đồng thời có sự khảo sát để có trải nghiệm trực tiếp => thảo luận về ý nghĩa của data, giao diện người dùng… và ghi lại biên bản các cuộc họp

- Bạn cũng cần lên danh sách các yêu cầu phi chức năng như về hiệu năng và bảo mật

### Architecture

- Như đã thảo luận ở chapter trước, bạn sẽ cần phải xác định Data flow architecture và system architecture náo sẽ được dùng

🡺 Sẽ là rất quan trọng khi có một người đã làm việc này trong thời gian dài, và nếu doanh nghiệp hiện không có ai có kinh nghiệm thì hãy đi tuyển người ngay nhé

🡺 Đây là critical step => nếu mà kiến trúc không đúng thì bạn sẽ phải thiết kế lại mọi thứ từ và bắt đầu lại từ đầu project này

### Design

- Bạn cần phải thiết kế ba phần chính của Data warehouse system là:

* Data stores
* ETL system
* Front-end applications

- Có nhiều skill được yêu cầu nên sẽ cần phải có nhiều người tham gia vào công đoạn này, ngoài ra bạn cũng cần phải thiết kết hai phất khác là:

* Data quality system
* Metadata

### Development

- Bạn cần phải build ba phần mà bạn đã design ở trên

- Với những cảnh báo và xem xét thì cả ba phần này nên được tiến hành xây dựng song song => Sự xem xét quan trọng nhất ở đây khi xây dựng song song đó là định nghĩa giao diện chính xác giữa các phần

### Testing

🡺 Đây được xem là điểm yếu chí mạng của cả dự án khi thường xuyên là công đoạn gặp nhiều vấn đề nhất trong phương pháp waterfall

- Khi mà quá trình phát triển kết thúc và công việc kiểm thử bắt đầu thì sẽ có rất nhiều “lần đầu tiên” xảy ra nên những sự ngạc nhiên là không thể tránh khỏi => Bạn sẽ bỏ ra rất nhiều tiền và thời gian để sửa chữa các vấn đề => Có thể giảm nhẹ những vấn đề bằng cách chạy toàn bộ kiến trúc một vài lần trong quá trình development hoặc tạo ra các bản mẫu để có thể phát hiện sớm vấn đề

### Deployment

- Một khi hệ thống đã sẵn sàng, lần đầu data được tải lên từ source system, tiến hành một vài test cùng với production environment để đảm bảo Data Warehouse đã được xử lý chính xác và front-end application đang sản xuất ra những số liệu đúng => Bạn sẽ tiến hành làm user guide, operation guide, troubleshooting guide cho team vận hành

- Bạn sẽ train users và team vận hành, hỗ trợ hệ thống, users và team vận hành trong một vài tuần, sau đó bàn giao toàn bộ hệ thống cho team vận hành, cuối cùng là kết thúc project

### Operation

- Users tiếp tục sử dụng Data Warehouse và ứng dụng

- Team vận hành tiếp tục quản trị Data Warehouse và hỗ trợ cho người dùng => Có ba dạng hỗ trợ cơ bản:

* Xử lý lỗi hoặc vấn đề xảy ra khi sử dụng hệ thống
* Quản trị người dùng mới và việc truy cập đúng của họ cũng như cả người dùng hiện tại
* Giúp đỡ người dùng sử dụng hệ thống

- Users sẽ luôn có những yêu cầu nâng cao như muốn thêm nhiều data hơn vào Data Warehouse, thêm tính năng vào front-end applications hay tạo ra các báo cáo mới hay cube mới => Team vận hành sẽ xem xét độ ưu tiên các yêu cầu và tiến hành thực hiện chúng

### Infrastructure setup

- Một trong những nhiệm vụ quan trọng nhất khi build một ứng dụng đó là chuẩn bị production environment nơi mà bạn sẽ tiến hành chạy ứng dụng và build development and testing environment

- This task consists of creating system architecture, doing technical design, procuring goods (buying the hardware and software), installing hardware, installing and configuring software, connecting networks (including firewall changes), testing the infrastructure, producing infrastructure documentation => Bàn giao toàn bộ cho team vận hành 🡺 Đây có thể coi là một dự án và cần được quản lý như một dự án

- Thường thì mọi người đánh giá thấp phạm vi ảnh hưởng của task này => dẫn đến Data Warehouse project bị delay bởi infrastructure chưa sẵn sàng

### Project management

- Khi bạn duy trì tiến độ kế hoạch dự án (a.k.a bạn biết được trạng thái của từng nhiệm vụ, ai sẽ làm chúng và khi nào làm)

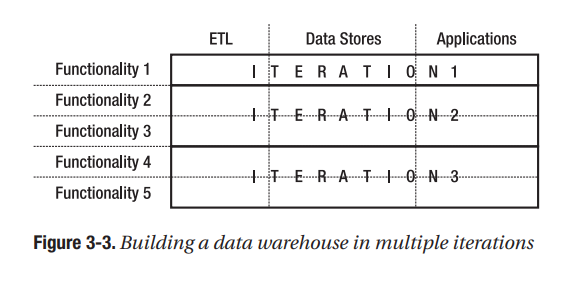
- Bạn sẽ cần phải giữ được kết nối đủ tố với toàn bộ stakeholder gồm nhà tài trợ dự án, người dùng và nhà phát triển, bằng các báo cáo trạng thái, các cuộc gặp mặt và xử lý các rủi ro một cách tích cực (thay vì đợi nó trở thành một mối nguy mới tiến hành giải quyết)

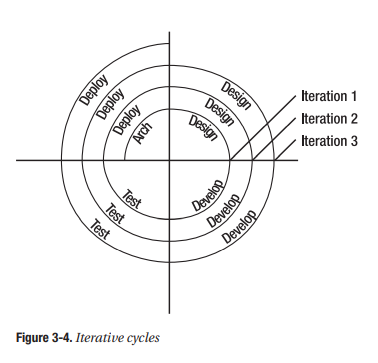
## Iterative methodology

- Nguyên tắc cơ bản của iterative methodology đó là phát hành các phần lớn hơn của dự án hoàn chỉnh để phát hiện sớm các vấn đề và thực hiện các điều chỉnh cần thiết khi người dùng không thể xác định đầy đủ các yêu cầu

- Nếu như bạn chỉ go live được một lần khi tiến hành công đoạn testing với waterfall thì ở đây bạn sẽ phát hành các phiên bản của hệ thống với production được chọn để xây dựng cho một số lượng user nhất định sử dụng để thu thập lại các phản hồi

- Ở mỗi vòng lặp thì bạn sẽ phải làm từ đầu đến cuối, tức ở mỗi vòng lặp bạn sẽ làm toàn bộ quá trình từ requirement, design, build, test và lựa chọn ra user sẽ sử dụng chúng, và bạn sẽ build toàn bộ các thành phần cần thiết để hỗ trợ cho phần production đó





- Lý do chỉnh để tiến hành project theo phương pháp này đó là giảm thiểu tối đa rủi ro. Bởi ngay từ vòng lặp đầu tiên các vấn đề đã được loại bỏ và đến vòng lặp cuối thì hệ thống đã vận hành trơn tru hơn rất nhiều

🡺 Ưu điểm của việc sử dụng phương pháp phát triển lặp quá rõ ràng là giảm thiểu rủi ro, và dự án sẽ có khả năng thành công cao hơn và delivered system sẽ có chất lượng tốt hơn

🡺 Nhược điểm chính là việc cơ sở hạ tầng cần được thiết lập và chuyển giao trước

# Functional and Nonfunctional Requirements

- Functional requirements xác định hệ thống sẽ làm gì, bao gồm các tính năng mà Data Warehouse cần có

- Nonfunctional requirement hướng dẫn và tạo rằng buộc cho kiến trúc, bao gồm về security, availability, và performance

## Identifying business areas

- Ví dụ về case study

## Understanding business operations

- Ở mỗi khu vực kinh doanh thì cần có sử hiểu biết về các quá trình, quy định và vấn đề của khu vực đó => Chúng ta sẽ cần tìm hiểu về business event, status, levels và roles:

* Một event là hoạt động diễn ra lặp lại mỗi giây mỗi phút hoặc có thể mỗi giờ mỗi ngày => một event có thể là một hành động mua hàng, đặt một món hàng…
* Một status là điều kiện của một đối tượng tại một thời điểm cụ thể => ví dụ: tình trạng của bài hát có thể là active, obsolete or upcoming
* Một level là quantitative measurement của một đối tượng trong một thời điểm cụ thể
* Roles sẽ là về ai/đối tượng nào/cái gì tham gia vào một sự kiện => Ví dụ là roles của purchase order event sẽ là nhà cung cấp, quản lý tài khoản, sản phẩm

## Defining functional requirements

- Functional requirements sẽ là những tính năng và chức năng có trong Data Warehouse, theo cách khác thì nó sẽ trả lời cho câu hỏi là hệ thống làm những công việc gì

- Thường những chức năng và tính năng sẽ có độ ưu tiên cần được thực hiện nên ở mỗi hệ thống sẽ cần phải phân cấp độ ưu tiên của các yêu cầu về chức năng

- Chúng ta review các yêu cầu chức năng, viết tài liệu và lấy được sign-off cần thiết

=> Review các yêu cầu với các điều khoản về sự hoàn chỉnh, tính khả thi và sự khả dụng của data

=> Viết tài liệu trong form của các use case

=> Business users review các use case và đăng nhập vào chúng

## Defining nonfunctional requirements

- Để có thể tìm ra các yêu cầu phi chức năng có sự thảo luận với team kiến trúc IT, team vận hành và business users

- Các yêu cầu phi chức năng cung cấp hướng dẫn và các hạn chế rằng buộc đối với kiến trúc hệ thống

- Các yêu cầu này có liên quan tới quản lý dự án và dự án sẽ được tiến hành ra sao => ví dụ như các yêu cầu liên quan đén hệ thống, tài nguyên, vị trí và ngân sách

- Cũng giống như yêu cầu chức năng, những người liên quan sẽ tiến hành review, viết tài liệu và lấy ra các sign-off cần thiết, review các yêu cầu dựa trên các điều khoản về sự hoàn chỉnh, tính khá thi và ảnh hưởng lên kiến trúc hệ thống

=> Viết tài liệu về yêu cầu in supplementary specificaiton format với những phần về khả năng sử dụng, ,khả năng tin tưởng, hiệu suất, khả năng hỗ trợ, rằng buộc thiết kế, tài liệu người dùng, help, giao diện, tiêu chuẩn ứng dụng

=> Business users, nhóm kiến trúc IT, team vận hành sẽ review tài liệu và đăng nhập vào chúng

## Conducting a data feasibility study

- Một data feasibility study là quá trình đi khám phá source system, hiểu được dữ liệu bằng cách liệt kê các rủi ro về dữ liệu chính và xác minh chúng, xác định xem có thể thực hiện dự án khả thi hay không

- Khám phá source system là đi giám sát về database platform, về cấu trúc csdl và truy vấn các bảng

- Hiểu được về data có nghĩa là tìm ra được data nằm ở đâu cho mỗi yêu cầu chức năng và hiệu được ý nghĩa cũng như chất lượng của data đó

🡺 Mục đích của việc làm một data feasibility study là lên ý tưởng về việc có hay không rủi ro về data nào đó khiến cho dự án thất bại, rủi ro về data bao gồm các rủi ro liên quan tới tính khả dụng và khả năng truy cập vào data

- Việc xác minh các rủi ro và giảm thiểu chúng là điều rất quan trọng

## Summary

- Với chương này, chúng ta đã định nghĩa các yêu cầu về chức năng cho case study, đồng thời là yêu cầu phi chức năng, cùng với đó là việc xác minh data và source system hỗ trợ những yêu cầu này. Điều đó này có nghĩa là chúng ta sẽ không phải gặp những rủi ro đối với dự án và hạn chế việc phải di chuyển tới Data Warehouse

- Ở chương tiếp theo, chúng ta sẽ tìm hiểu về việc thiết kế một data model như nào để có thể làm hài lòng những yêu cầu trên

# Data Modeling

- Với case study chính thì chúng ta sẽ sử dụng NDS + DDS architecture để xây dựng để sẽ có trải nghiệm về việc thiết kế store chuẩn hoá và store về chiều

- Bắt đầu bằng việc tìm hiểu business requirements và rồi thiết kế DDS phù hợp, sau đó định nghĩa các thuộc tính fact và dim, đồng thời là data hirearchy

- Sau đó chúng ta sẽ ánh xạ data trong DDS với source system, nghĩa là xác định nguồn gốc của các cột trong bản dim và fact

- Sau đó chúng ta sẽ thiết kế NDS bằng cách chuẩn hoá DDS và giám sát data từ source system mà chúng ta đã ánh xạ

## Designing the dimensional data store

- Thông thường sẽ có 6 lĩnh vực trong kinh doanh của một doanh nghiệp, gồm:

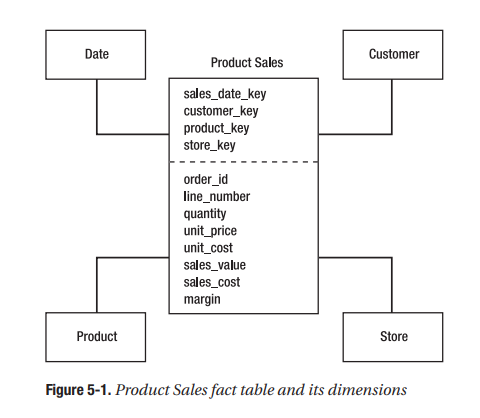
* Product sales
* Subscription sales
* Subcriber profitability
* Supplier performance
* CRM campaign segmentation
* CRM campaign result

- Bắt đầu với product sales thì một product sales event diễn ra khi có một khách hàng mua một món đồ

=> Roles sẽ là customer, store và product

=> Levels (or measures) sẽ là quantity, unit price, value, direct unit cost và undirect unit cost

- Chúng ta sẽ đặt các measure vào trong bảng fact còn các role sẽ tạo thành các bảm dim tương ứng, như vậy thì business event sẽ trở thành các dòng của bảng fact



- Trong bảng fact sẽ có quantity, unit\_price và unit\_cost bắt nguồn từ source system nhưng sales\_value, sales\_cost và margin sẽ được tính toán

- 4 khoá ngoại của bảng fact product sales tham chiếu đến 4 khoá chính tương ứng ở 4 bảng dim

- Bên cạnh đó, việc khai báo grain của bảng fact là điều quan trọng => Grain là đơn vị nhỏ nhất của sự xảy ra của business event nơi mà event được đo lường => Có thể hiểu rằng là grain sẽ câu trả lời cho thông tin một dòng trong bảng fact tương ứng với cái gì.

=> Với product sales event thì grain sẽ là từng món đồ được bán

- Các rule chung sẽ thường gắn liền với source system => Chúng ta cần tái tạo và bắt chước logic của source system => Trong mọi trường hợp, hãy luôn trao đổi với business people trước về business logic bởi họ hiểu về luật business => Sẽ là khôn ngoan khi luôn xác nhận được data model và business logic cùng với source system => đảm bảo rằng output của Data Warehouse phản ánh đúng điều kiện kinh doanh

- Có 2 cột order\_id và line\_number trong bảng fact là degenerate dimensions => Là một chiều với chỉ một thuộc tính nên vì thế nó được đẩy vào trong bảng fact

- First timestamp column sẽ lưu thời gian record được load lên Data Warehouse, trong khi second timestamp column sẽ lưu thời gian gần nhất mà record được update

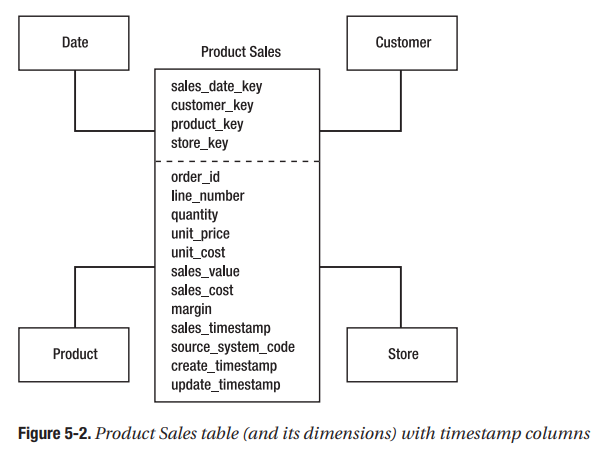
- Thông thường các degenerate dimension sẽ được chọn ra 1 tổ hợp để làm primary key(s) cho bảng fact

- Sự tập hợp của một bảng fact và các bảng dim xoay xung quanh chúng được gọi là một **data mart**

=> Có rất nhiều dạng phức tạp hơn của data mart khi có thể có nhiều hơn một bảng fact

- Định nghĩa về data mart => Là một nhóm các bảng fact có liên quan và các bảng dim tương ứng chứa các đo lường, ước tình của business events, được phân chia danh mục bởi các chiều

- Data mart tồn tại trong DDS và KHÔNG tồn tại trong NDS => Chỉ ứng dụng được nếu như Data Warehouse ở trong một dimensional model



- Chúng ta sẽ tiến hành định nghĩa data type cho từng cột, sau đó thêm description cho từng cột và giá trị ví dụ của từng cột

## Dimension tables

- Một dimension table là một bảng chứa các thuộc tính khác nhau giải thích dimension key trong bảng fact, cụ thể chúng lý giải về các điều kiện của thực thể tại thời điểm mà business event diễn ra

- Chúng ta sẽ có một định nghĩa đó là toàn vẹn tham chiếu => đây là một concept của việc xây dựng mối quan hệ cha con giữa hai bảng. Ở đây khoá chính sẽ nằm ở các bảng dim, bảng fact sẽ có các khoá ngoài tham chiếu đến khoá chính của các bản dim

- Bảng dim chứa các thuộc tính giải thích cho các điều kiện của các thực thể tham gia vào bussiness event lưu ở trong bảng fact, chúng được lưu trữ ở trong các bảng chiều dưới dạng các cột => dimensional attributes

## Date dimension

- Hầu hết các data mart riêng lẻ đều có date dimension nên việc model date dimension một cách chính xác là rất quan trọng => các thuộc tính nằm trong data dimension đều được yêu cầu từ tất cả các bảng fact và data mart

- Các cột trong date dimension được chia thành bốn nhóm:

* Date format => chứa dates ở nhiều dạng khác nhau
* Calendar date attributes => chứa các đối tượng của một date như ngày, tháng, năm
* Fiscal attributes => chứa các đối tượng liên quan đến lịch tài khoá như fiscal week, fiscal period và fiscal year
* Indicator columns => chứa các giá trị bool, dùng để xác định các date cụ thể có thoả mãn điều kiện cụ thể hay không

## Slowly changing dimension

- SCD là kĩ nghệ được sử dụng để lưu trữ giá trị trong lịch sử của dimension attributes

- Có ba thể loại SCD:

* SCD loại 1 sẽ overwrite lại các giá trị cũ => các giá trị cũ sẽ không được giữ lại
* SCD loại 2 sẽ giữ lại giá trị cũ và thêm giá trị mới bằng một dòng
* SCD loại 3 sẽ giữ lại giá trị cũ và thêm giá trị bằng một cột

=> Ở đây SCD loại 2 sẽ là linh hoạt nhất cho việc lưu trữ giá trị lịch sử của thuộc tính về chiều khi có thể lưu trữ rất nhiều version cũ thay vì thay đổi cấu trúc của bảng

=> Với SCD loại 3 thì sẽ phù hợp khi bạn biết rõ rằng các thuộc tính không thay đổi quá nhiều (<= 5) và có thể biết được số lượng các version, bên cạnh đó là khi số lượng các thuộc tính thay đổi đồng thời là rất lớn

- Vậy việc chậm ở đây là như nào?

=> Nếu việc thay đổi của các thuộc tính chiều diễn ra mỗi 1 quý hoặc chậm hơn => Slowly changing

=> Nếu việc thay đổi của các thuộc tính chiều diễn ra mỗi tháng hoặc nhanh hơn => Rapidly changing

## Product, customer, store dimensions

- Ví dụ của case study

## Subscription sales data mart

- Với những gì đã làm product sales data mart thì có thể làm các data mart khác theo cách giống nhau, bao gồm:

* Bắt đầu với bảng fact
* Xác định các dimensional key column trong bảng fact
* Đặt grain cho bảng fact
* Tạo các bảng dim và take into account the SCD

=> Nếu các bảng dim ở trong data mart mới đã tồn tại thì không cần phải tạo mới mà có thể tái sử dụng các dimension hiện tại

=> Nếu các thuộc tính mà data mart lại không tồn tại ở trong các dimension thì sẽ thêm mới vào các dimension hiện tại

- Một data mart (gồm bảng fact và các bảng dim của chúng) lưu trữ một tập hợp của business event trong một business area cụ thể => Để thiết kế các bảng fact và bảng dim thì sẽ cần có kiến thức về kinh doanh trong business area

- Có một số điều cần được làm rõ sau:

1. Với lý do chuyên môn hay sự đơn giản và nhất quán về thiết kế thì việc sử dụng star schema sẽ được ưu tiên hơn hẳn so với snowflake schema
2. Nếu dimension đã tồn tại thì sẽ tái sử dụng chính thay vì tạo ra một phiên bản khác
3. Sẽ phải có sự nhất quán với những định nghĩa về data

- Với data mart này thì sẽ cần phải tạo một periodic snapshot fact table => lưu các state hay condition của business ở một thời điểm kịp lúc => Để thiết kế được một periodic snapshot fact table thì

1. Phải xác định được period
2. Phải xác định về measures, dựa vào data nào users muốn có để phân tích
3. Cần phải xác định các bảng dim cho bảng fact => cần tìm ra những gì users cần để phân tích measure
4. Cần xác định grain cho bảng fact

## Supplier performance data mart

## CRM data mart

## Data hierarchy

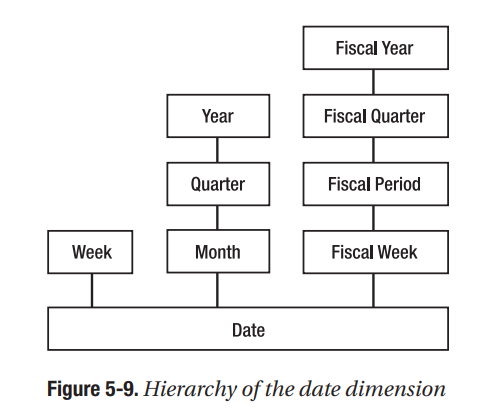
- Trong các bảng dim sẽ có một cấu trúc nhất định tên là hierarchy (sự phân cấp hay thừa kế) => rất quan trọng bởi nó cung cấp cho bạn những con đường để có thể đi sâu vào việc phân tích dữ liệu

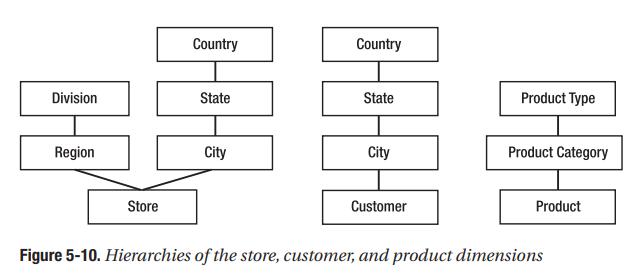
- Trong các bảng dim, đôi lúc một thuộc tính hay cột sẽ là subset của thuộc tính khác => giá trị một thuộc tính có thể được tạo ra từ các thuộc tính khác

=> thuộc tính có thể được dùng để tạo ra là higher level

🡺 Các level trong bảng dim gọi là dimensional hierarchy

- Ví dụ về phân cấp các thuộc tính của product sales fact table với 4 bảng dim như sau:





- Để áp dụng hệ thống phân cấp trong trường hợp của riêng của mỗi công ty hay case study, trước tiên hãy xem các cột (thuộc tính) trong bảng thứ nguyên để tìm xem có bất kỳ nhóm hoặc tập hợp con nào không. Sắp xếp các thuộc tính theo các cấp độ phù hợp, nghĩa là bạn cần đặt thuộc tính cấp cao hơn lên trên thuộc tính cấp thấp hơn => Xác định xem có nhiều đường dẫn (nhánh) trong cấu trúc phân cấp hay không; ví dụ, cửa hàng trong Hình 5-10 có thể được nhóm theo cả thành phố và khu vực. Bạn có thể làm tất cả điều này bằng cách sử dụng công cụ cấu hình dữ liệu. Hệ thống phân cấp là một phần tích hợp của thiết kế thứ nguyên

## Source system mapping

- Source system mapping là một hoạt động ánh xạ DDS tới source system

- Sau khi hoàn thành thiết kế DDS, bước tiếp theo đó là ánh xạ từng cột ở trong DDS tới source system để có thể biết được data mà chúng ta đặt vào trong các cột đến từ đâu => cần xác định transformation và calculations được yêu cầu đến, lấy từ cột nguồn đến cột đích

- Sẽ là cần thiết khi có thể hiểu được từng hàm mà logic ETL phải tính toán cho từng cột trong DDS, bởi một cột trong DDS có thể đến từ nhiều bảng trong source system và ODS lại tích hợp data từ nhiều nguồn

- Chúng ta sẽ cần viết các tên bảng của source system dưới dạng viết tắt, viết các dạng chuẩn để có thể tiến hành các phép nối giữa các bảng source system, ánh xạ các cột dimension tới bảng và cột của source system, sau đó tiến hành viết các chuyển đổi

=> Tất cả các bảng fact và bảng dim ở trong các data mart khác cũng sẽ làm như vậy

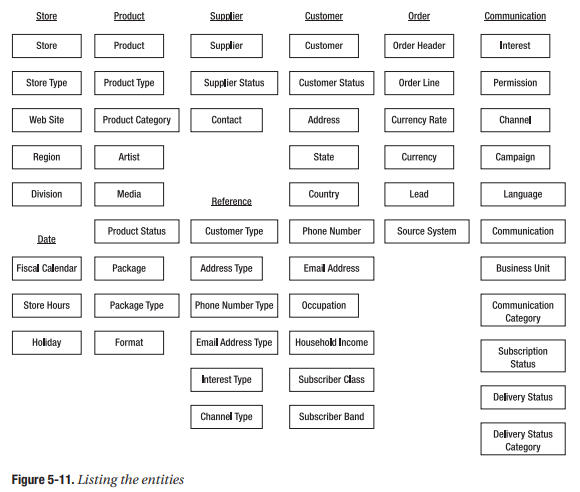
## Designing the normalized data store

- NDS là data store nằm ở giữa stage và DDS

- Trong kiến trúc này NDS sẽ là master data store, chứa các data set hoàn chính, bao gồm toàn bộ các version của transaction data cũng như master data

- NDS gồm các transaction table và master table, trong đó transaction tables chứa các business transaction hoặc business event còn master table sẽ chứa các những đối tượng tham gia vào business event

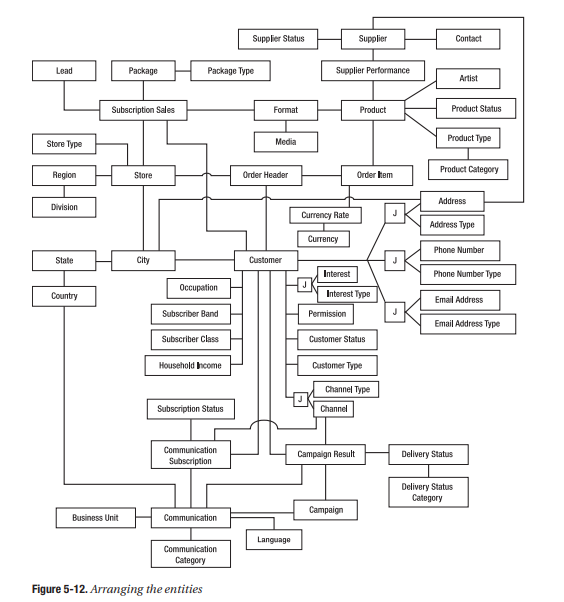
- Đầu tiên, ta cần phải lên danh sách các thực thể dựa vào bảng nguồn và các thuộc tính fact và dim trong DDS, lên danh sách tất cả các bảng trong source system mà đã được định dạng trong suốt quá trình ánh xạ source system



- Tiếp theo, ta chuẩn hoá bảng dim và fact của DDS thành một list các bảng chuẩn hoá được phân chia, sai đó tiến hành combine 2 list bằng cách loại bỏ các bảng trùng và phân chia cúng và các subject area khác nhau

- Một số nhóm các bảng sẽ được phân mục vào các business area, các bảng liên quan đến nhau sẽ được cho vào chung một group

- Sau đó, ta sắp xếp các thực thể dựa theo mối quan hệ của chúng để cho phép chúng ta có thể xây dựng được việc toàn vẹn tham chiếu giữa các thực thể, bằng cách kết nối bảng con với bảng cha, với khoá ngoại ở bảng con tham chiếu đến khoá chính ở bảng cha



=>> Bảng fact trong DDS trở thành bảng con trong NDS còn bảng dim trong DDS trở thành bảng cha trong NDS

- Các J boxes là các bảng junction cho phép các mối quan hệ M-N, chúng ta gọi là nó là các bảng cầu nối

🡺 Khi thiết kế NDS, ta cần phải tuân thủ luật chuẩn hoá, mục đích là để giảm thiểu data dư thừa và làm cho việc có thể duy trì data được dễ dàng hơn => Việc chọn form chuẩn hoá có thể ảnh hưởng trực tiếp đến hiệu suất => tuỳ vào case mà lựa chọn, có thể là 2NF, 3NF hay BCNF

- Bước tiếp đó lên danh sách các cột, ở đây chúng ta sẽ lấy các cột của DDS và source system nhưng KHÔNG phải lấy toàn bộ chúng mà chỉ chọn ra các cột liên quan

- Sau đó, chúng ta lấy datatypes từ source system => Nếu một cột được lấy từ 2 source system khác nhau, hãy lấy cái có khả năng lưu trữ giá trị lớn nhất (longest one)

=>> Then we do the same thing with all the NDS tables; in other words, we list their columns. The next step is to write down the source and transformation. This should not be difficult because we defined them when we did the source system mapping in the DDS.

🡺 Now the NDS is complete. It has all the tables necessary to create the DDS. It is highly normalized. We have defined all columns in all tables, along with the source and transformation for each column. We can now return to the DDS and redefine the source for each DDS column in the NDS. This is required for building the ETL between NDS and DDS. We can do this by identifying the source table in the NDS for each DDS table. Then we identify the corresponding NDS column that contains the source data for each DDS column.

## Summary

- Ở chương này, chúng ta đã thiết kế được DDS và NDS của hệ thống trong case study, bắt đầu từ yêu cầu về kinh doanh và mô hình DDS

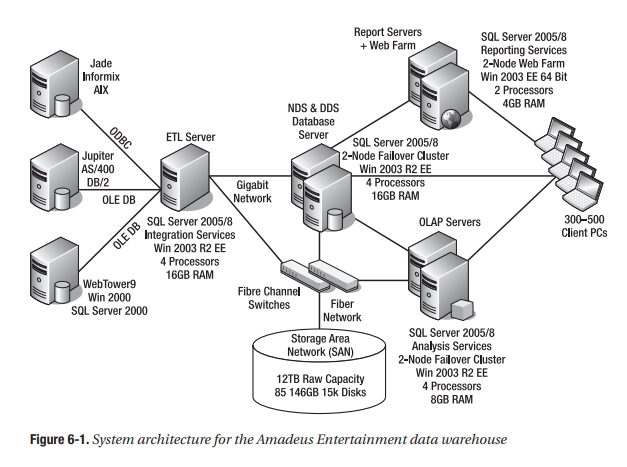
- Chúng ta đã định nghĩa về các bảng fact và dim, tiến hành ánh xạ source system, nơi mà chúng ta xác định nguồn gốc của từng cột và các biển đổi được yêu cầu

- Sau đó, tiến hành thiết kế NDS, lên danh sách các thực thể và định nghĩa các mối quan hệ, tiếp đến là lên danh sách các cột được yêu cầu và định nghĩa nguồn và biến đổi => quay lại DDS để xác định lại nguồn gốc của từng cột trong NDS

# Physical Database Design

- Ở chương này chúng ta sẽ trao đổi về các vấn đề như hardware platform, storage cosiderations, partitioning và indexes

## Hardware platform



- Thông thường, với các yêu cầu phi chức năng thì một Data Warehouse sẽ phải luôn khả dụng 24 giờ mỗi ngày, 7 ngày mỗi tuần và downtime chỉ được phép <= 1 tiếng trong 1 tháng

=> Với các doanh nghiệp mà sử dụng Data Warehouse hỗ trợ hoạt động CRM, bao gồm customer analytics và campaign management operation, việc có thể để cho Data Warehouse luôn hoạt động là rất quan trọng bởi nó hỗ trợ cho dòng doanh thu được sinh ra từ hoạt động CRM

=> Từ đây chúng ta có xu hướng sẽ cài đặt một hệ thống là các cluster có thể hoạt động luân phiên, ở đây là việc cấu hình các thành phần thành các server (node) giống hệt nhau => Khi một node không khả dụng thì ngay lập tức một node khác sẽ được thay thế để tiến hành chạy

- Ở trong hệ thống này thì Reporting services, Analysis services sẽ được thiết kế trong một cluster có sự thay đổi luân phiên, còn SSRS thì được triển khai trong một network load balanced (NLB) cluster

- SSIS thì không phải là cluster-aware application bởi SSIS không có yêu cầu phải luôn khả dụng 24/7 đối với end-users nhưng nếu SSIS dừng trong một khoảng thời gian thì sẽ ảnh hưởng tới data currency

- Những sự xem xét về network cũng sẽ cần được đề cập khi cấu hình tốc độ mạng giữa các thành phần khi mà chúng làm tăng tỉ lệ data-loading từ nguồn đến đích

🡺 Câu hỏi tiếp theo sẽ liên quan đến server technical specification, với những câu hỏi về memory, số lượng CPU, cần bao nhiêu không gian bộ nhớ, phiên bản của Windows Server hay SQL Server

- Với một ETL server thì bạn sẽ cần phải xác định độ phức tạp transformation logic, lượng bộ nhớ yêu cầu, số luồng, lõi của CPU, xung nhịp, tốc độ đọc ghi => Bạn sẽ phải để ý đến sự phát triển trong tương lai của hệ thống khi phải ước lượng một Data Warehouse mới hay một dự án ETL với yêu cầu nâng cao hơn có thể diễn ra

- Với OLAP server cluster hardware thì đó là số lượng các large cube mà có thể chạy trên Analysis Services

- Với database server thì đây có thể là phần khó nhất để tính toán vì nó hầu như phụ thuộc vào các ứng dụng truy cập vào database có tần suất và yêu cầu truy vấn có lớn hay không => Dưới đây là những xem xét khi ước tính các số liệu để có thể hoạt động hiệu quả:

* Số lượng và sự phức tạp của các reports, applications và direct queries hitting the DDS
* Chúng ta sẽ lượng chọn phương pháp ETL hay ELT cho việc phân bổ vào NDS và ODS
* Việc tính toán từ stage tới NDS/ODS và độ phức tạp của quy định tường lửa
* Số lượng và size của data stores
* Data stores được thiết kế như thế nào (đánh chỉ mực, phân hạch…)
* Số lượng các database khác được host trên cùng một server và trong tương lai

- Sau khi xác định được việc cấu hình server cho SSIS, SSRS, SSAS và database server thì chúng ta sẽ cần xác định hệ thống vận hành nào và phiên bản SQL Server cần thiết cũng như license model phù hợp nhất với kiến trúc

## Storage considerations

- Thứ tiếp theo mà chúng ta cần có cho cơ sở hạ tầng sẽ là disk space

- Sau khi đã hoàn thành việc thiết kế DDS và NDS => chúng ta có thể tính toán được lượng bộ nhớ thích hợp bằng cách tính toán dộ lớn của các bảng fact và dime để lấy được size của DDS. Với NDS, chúng ta ước lượng được dựa theo row size của số lượng các dòng của từng bảng

- Sau đó sẽ ước tính size của stage database dựa theo source system tables và data extraction method

- Với các bảng fact, cần tính thêm % số lượng bytes để cung cấp thêm các cột chúng ta có thể thêm trong tương lai, % số giao dịch có thể thêm trong tương lai

=>> Chúng ta sẽ làm tương tự với mọi bảng fact và bảng dim => Chúng ta sẽ tính được size của toàn bộ các bảng trong DDS

=>> Tương tự với NDS, tính toán row size với data type của từng column, sau đó ước lượng số lượng dòng thông qua truy vấn vào source system => Số lượng dòng ở trong bảng NDS phải tương đương với số lượng dòng ở trong source system và phải lưu ý có sự overlap giữa các source system (các dòng giống nhau ở các source system)

=>> Với stage database thì chúng ta cần lên danh sách các bảng nguồn cần để tiến hành ánh xạ vào source system. Với mỗi bảng, tiến hành truy vấn để lấy được số lượng dòng => Nếu phương pháp trích xuất tới stage là gia tăng thì hãy đếm có bao nhièu dòng được thêm or update mỗi ngày trong source system

- Bạn sẽ chỉ cần lấy các cột từ trong bảng source system chứ không phải toàn bộ các cột, đồng thời bạn có thể sẽ cần lưu trữ các giá trị này trong 3-5 ngày phụ thuộc vào chiến lược backup trong trường hợp bạn cần phải reload chúng lên NDS vì một vài lý do nào đó

=>> Với metadata thì không cần quá nhiều, lưu trữ 7 dạng metadata: data definition and mapping metadata, data structure metadata, source system metadata, ETL processes metadata, data quality metadata, audit metadata, and usage metadata

=>> Bạn cũng sẽ cần phải phân bổ bộ nhớ cho 2-3 DDS nhỏ cho mục đích data mining hay specific analysis

- Ngoài ra bạn sẽ cần phân bổ bộ nhớ cho chỉ mục và các RDBMS overhead như table header, row ID, block header và trailer, fill factor, page splits, các cơ chế SQL Server internal DBMS khác

## Configuring database

- Ví dụ về case study

- Với stage database, chúng ta không cần việc update thống kê tự động khi không tiến hành đánh chỉ mục các bảng, tuy nhiên vẫn cần simple recovery model

- Hãy nhớ đặt recovery mode to full cho metadata database, bởi size của metadata database là nhỏ và cách chúng ta sử dụng chúng sẽ giống việc truy vấn dụng OLTP hơn là truy vấn Data Warehouse, tức là chúng thường xuyên được cập nhật

## Creating DDS database structure

- Sau khi đã tạo xong csdl thì chúng ta đã có thể tạo ra các bảng dựa trên các thiết kế logic đã được tạo ra ở chương trước

- Các dimension key của các bảng dim sẽ cần được khai báo như là primary key và clustered index => rất có ích cho hiệu suất truy vấn

- Chúng ta sẽ cần phải viết một đoạn code để check và drop các ràng buộc vể khoá ngoại trước khi drop và tạo bảng

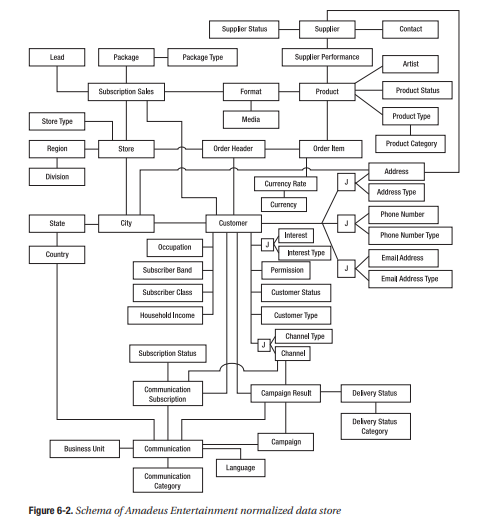
- Nếu chúng ta đặt bảng fact và bảng dim trong các disk khách nhau thì sẽ làm giảm disk contention và từ đó tăng hiệu suất truy vấn, phụ thuộc vào các bảng dim join vào bảng fact có rõ ràng không trong các truy vấn

- Với các khoá ngoại ở bảng fact, những người tối ưu việc truy vấn sẽ sử dụng các ràng buộc về khoá chính và khoá ngoại để build một kế hoạch khai thác các truy vấn với hiệu năng cao

- Sau khi đã hoàn thành toàn bộ các bảng fact và dim, chúng ta sẽ tạo một batch file sử dụng sqlcmd được gọi các script lần lượt

## Creating the NDS

- NDS sẽ có một chút khác biệt so với DDS, và sự khác biệt chính là việc chuẩn hoá



- Các bảng của NDS có thể phân mục thành 3 nhóm:

* Transaction tables => chứa các business event và bắt nguồn từ các bảng fact DDS
* Master tables => chứa các thực thể business và bắt nguồn từ các bảng dim DDS
* Attribute tables => chứa các thuộc tính của các thực thể business và bắt nguồn từ các cột dimension attribute

- Giống như với DDS thì chúng ta sẽ cần đặt các bảng và các filegroup khác nhau nằm ở các ổ đĩa vật lý khác nhau. Với NDS thì thông thường sẽ chia chúng theo subject areas

- Với indexes thì chúng ta sẽ để riêng so với các bảng trong ổ đĩa với lý do về hiệu suất

- Khi chúng ta muốn tìm xem liệu một foreign key có đang tồn tại hay không thì bạn cần phải cụ thể xem bảng nào mà có foreign key đó => Việc này rất quan trọng để đảm bảo các foreign key thuộc về đúng bảng của nó, khi việc 2 foreign key của 2 bảng khác nhau có tên giống nhau là hoàn toàn có thể

- Ngoài ra, việc đặt tên cho primary key và foreign key cần có một quy ước nhất quán

- Mối quan hệ giữa 3 loại bảng trong DDS như sau: bảng fact sẽ được kết nối tới bảng master, cùng với đó bảng thuộc tính sẽ miêu tả một thuộc tính cụ thể trong bảng master

🡺 Nhưng với NDS thì không như vậy, một bảng fact ở trong DDS có thể được chuẩn hoá tới 2 bảng transaction trong NDS, và một bảng dim có thể không liên kết trực tiếp đến các bảng thuộc tính

* Nếu quan hệ là M-N => cần một junction table hay bảng cầu nối => junction table chứa các khoá chính của cả bảng dim và bảng thuộc tính
* Nếu quan hệ là 1-N => kết nối trực tiếp với nhau

🡺 Chúng ta đã hoàn thành việc tạo ra NDS, giống như DDS thì chúng ta sẽ chỉ cần tạo một batch file chứa sqlcmd để gọi các script chúng ta đã code và phải cẩn thận với thứ tự mà bạn sẽ build các bảng => Vì sự toàn vẹn tham chiếu nên việc build chúng theo một thứ tự nhất định là cần thiết

## Using views

- View là một đối tượng csdl với các cột và dòng như một bảng nhưng lại không có ở trong ổ đĩa => Chúng ta sẽ dùng lệnh SELECT để tạo ra view với việc lọc các cột và dòng từ một hoặc vài bảng sử dụng lệnh JOIN

🡺 Trong Data Warehouse thì views được sử dụng cho ba mục đích sau:

* To create a conform dimensions in dimensional data stores => Conformed dimensions có nghĩa là hoặc chúng là những bảng giống nhau hoặc là bảng con của cái còn lại
* To shield users from physical tables, making it simpler for the users as well as to restrict access => View tạo ra một layer ảo ở phía trên bảng vật lý. Users không truy cập vào bảng vật lý mà thay vào đó họ truy cập và views. Layer được thêm nào cho phép chúng ta tạo ra một bảng ảo lấy data từ một vài bảng và cũng hạn chế users truy cập được vào chi tiết các cột và dòng
* To increase the availability, particularly to make the data warehouse up and running when we are populating it => Được dùng trong các user-facing data stores như DDS, ODS. Với mỗi bảng dim và fact thì chúng ta sẽ tạo 2 bảng vật lý và 1 view sẽ select từ 1 trong 2 bảng vật lý

### To create a conform dimension

- Đôi lúc khi tạo một data mart thì các chiều chúng ta cần đã tồn tại nhưng độ chi tiết là không như chúng ta muốn, chỉ vài cột là đáp ứng => Tiến hành tạo một view từ các dimension đã tồn tại và select như cột và dòng chi tiết theo mong muốn

- Khi chúng ta có conformed dimensions giữa 2 data mart thì chúng ta có thể đào sâu vào phân tích từ mart này sang mart kia => Lợi ích ở đây đó là việc chúng ta không phải điền các dimension với grain cao hơn, bởi khi data ở trong underlying dimension thay đổi thì nó cũng tự động thay đổi

### To create a virtual layer on top of physical tables to make it simpler for the users

- Có 3 dạng view chính hay được sử dụng đó là data mart view, last month view và specific store view, ngoài ra còn có thêm SCD type 2 and type 3 view => SCD chứa cả giá trị quá khứ và hiện tại và chúng ta sẽ dùng view để lựa chọn giá trị hiện tại mà thôi

- Data mart view là khi chúng ta tạo một view ở trên đầu bảng fact các bảng dim xoay quanh chúng, và giúp cho end users có thể truy cập một bảng đơn được lấy nguồn từ nhiều bảng vật lý, sử dụng câu lệnh SELECT và JOIN

- Với last month view thì việc tạo ra một view với mệnh đề WHERE sẽ giúp cho user rất nhiều để làm các báo cáo sau mỗi tháng

- Với specific store (or country, or branch, or locale, or region) sẽ là view khác mà thường xuyên được sử dụng

### To increase availability

- Khi bạn điền vào một bảng fact, thay đổi diễn ra và sẽ tiền hành khoá bảng khỏi việc truy cập => Có 2 mục đích cho việc locking này đó là để tăng tốc độ tải và tránh việc thất thoát dữ liệu

- Trong thực tế, bảng fact sẽ là view được xây dựng từ 2 bảng vật lý khác nhau

- Chúng ta có công nghệ cho việc intraday implementation như sau:

1. Set the view to point to table 2.

2. Load today’s data into table 1.

3. Set the view to point to table 1.

4. Load today’s data into table 2.

=> Bằng cách sử dụng công nghệ này thì Data Warehouse sẽ luôn được cập nhật, việc tạo view chỉ mất vài giây. Công nghệ này thường được sử dụng để tăng sự khả dụng của Data Warehouse, bao gồm packaged applications chạy trên Normailized Data Warehouse cũng như custom-developed Data Warehouse

## Summary tables

- Trong những thứ có thể cải thiện hiệu suất của Data Warehouse thì summary table đứng đầu trong danh sách, sau đó là table partitioning và indexing

- Hãy nhớ rằng một summary table sẽ cần được cập nhật mọi lúc mỗi khi mà underlying fact table được update, thời điểm lý tưởng nhất để refresh đó là sau khi việc điền được tiến hành của underlying fact table

- Chúng ta có 3 dạng của fact table là:

* Transactional fact table => phù hợp cho transaction fact table
* Periodic snapshot fact table => chúng ta có latest summary table, chứa các version ngày hiện tại của snapshot
* Accumalative snapshot fact table

## Partitioning

- Nó giúp cản thiện hiệu suất của Data Warehouse chỉ sau summary table, chúng ta có thể phân hạch được cả view và table

- Có 2 dạng partitioning:

* Vertical partitioning => chia cảng bảng theo chiều dọc thành các bảng nhỏ hơn, mỗi bảng chứa một vài cột của bảng gốc
* Horizontal partitioning => Chia bảng theo chiều ngang thành một vài bảng nhỏ hơn và mỗi bảng chứa một khoảng các dòng của bảng gốc

- Sự thuận theo tự nhiên của các bảng fact trong dimensional Data Warehouse đó là những nội dung sẽ được sắp xếp theo thời gian => Bởi đặc điểm tự nhiên đó mà sẽ là tốt nhất để tổ chức bảng theo date

- Đầu tiên chúng ta drop table, chức năng phân hạch, lược đồ phân hạch nếu có

- Chúng ta cụ thể hoá boundary values

- Sau khi tạo lược đồ phân hạch, chúng ta sẽ tạo các bảng bằng cách cụ thể hoá các lược đồ phân hạch với các cột đã được phân hạch trong brackets at the end of the DDL => để inform SQL Server mà chúng ta muốn phân hạch bảng sử dụng lược đồ phân hạch

- Khi tạo ra một bảng phân hạch, nếu các filegroup rằng buộc khác với filegroup bảng thì SQL Server ẽ làm bất kì thứ gì được cụ thể hoá trong filegroup rằng buộc và bỏ qua filegroup bảng

- Partition index là nơi tốt nhất trong các filegroup giống nhau cũng như bảng phân hạch vì chúng cho phép SQL Server để chuyển đổi các phân hạch được hiệu quả => các chỉ số được căn chỉnh với table => Cho phép chuyển đổi và di chuyển các phân hạch, và chỉ số phải được căn chỉnh cùng với bảng => Thực tế chúng ta không di chuyển data mà chúng ta thay đổi metadata only around where the data is

- Parition maintenance là hoạt động quản trị một bảng đã được phân hạch mà từ đây chúng ta có thể sẵn sàng cho việc tải data và truy vấn

## Indexes

- Việc dùng chỉ số có thể cải thiện khá nhiều hiệu suất truy vấn và tải của Data Warehouse

- Trong DDS, chúng ta có các bảng fact và dim, yêu cầu các chỉ số và primary key khác nhau

- Mỗi bảng dim sẽ có một cột surrogate key, đấy là các cột có quan hệ 1-1 và giá trị là unique => Chúng ta sẽ đặt cột này vào khoá chính của bảng dim và vào clustered index của bảng dim

- Với các bảng fact trong SQL Server Data Warehouse thì có 2 cách tiếp cận trong việc xác định khoá chính và clustered index:

* Create a fact table surrogate key column
* Select the minimum combination of columns that make a row unique as the primary key => chọn ra bộ siêu khoá tối thiểu

## Summary

- Đây là một chương khá là dài, bởi việc thiết kế cơ sở dữ liệu là cornerstone của Data Warehouse. Chúng ta sẽ xây dựng ETL và các ứng dụng trên nền tảng này, nên bắt buộc phải làm đúng. Trong chương này, chúng ta đã thảo luận chi tiết về nền tảng phần cứng và thiết kế hệ thống, tính toán không gian ô đĩa, việc tạo các csdl, table và view

- Chúng ta đã cover 3 yếu tố chính giúp cải thiện hiệu suất của data warehouse, đó là summary table, partitioning và indexing. Chúng ta cần đảm bảo rằng chúng được thiết lập đúng ngay từ đầu, khi chúng ta tạo databases

- Bây giờ chúng ta đã xây dựng được databases, ở 2 chương tiếp theo sẽ là việc làm thể nào để trích xuất data từ source system và điền vào NDS và DDS databases của chúng ta

# Data Extraction

- Chúng ta đã tạo được các bảng trong DNS và DDS, đây sẽ là lúc để điền vào chúng, tuy nhiên sẽ cần phải lấy data từ source system

## Introduction to ETL

- ETL là viết tắt của Extract, Transform and Load => Đây là quá trình lấy và biến đổi data từ source system và đẩy chúng lên Data Warehouse. ETL đã xuất hiện vài thập kỉ và đã liên tục phát triển và cải thiện kể từ sự hình thành

- Có một vài các nguyên tắc cơ bản cần phải hiểu khi mà trích xuất data từ một source system cho mục đích điền vào Data Warehouse:

1. Khối lượng data được lấy là rất lớn. Trong khi hệ thống OLTP được thiết kết với việc data chỉ được lấy với một lượng nhỏ hơn khá nhiều => cần phải cẩn thận nếu không sẽ làm chậm source system đi rất nhiều
2. Chúng ta cần việc trích xuất diễn ra nhanh nhất có thể, nhỏ nhất có thể, không thường xuyên diễn ra, và sự thay đổi trong các source system là ít nhất có thể => tốt nhất là không có sự thay đổi nào

=>> Khi trích xuất data từ một source system thì hãy cẩn thận hết sức và tránh gây ảnh hưởng đến source system quá nhiều

- Sau khi trích xuất data thì sẽ cần phải đẩy chúng lên Data Warehouse nhanh nhất có thể, lý tưởng nhất là đưa thẳng lên mà không lưu trữ chúng tạm thời ở đâu đó => Sẽ cần ứng dụng một vài biến đổi với data từ source system để có thể phù hợp với format và cấu trúc của data trong NDS và DDS, thông thường đó là formatting, standardization, lookup (tra cứu), aggregation

- Data đẩy lên Data Warehouse cần được dọn dẹp và có chất lượng tốt, phải tiến hành nhiều quá trình kiểm tra trước khi tải chúng lên warehouse

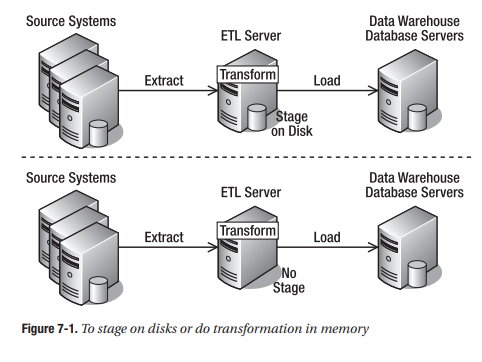
- Bên cạnh đó, có 2 nguyên tắc quan trọng khác đó là leakage và recoverability:

* Leakage diễn ra khi quá trình ETL nghĩ rằng nó đã tải hết các data từ các source system nhưng thực tế lại bị miss một vài record => Một quá trình ETL tốt sẽ không để miss một cái nào
* Recoverability có nghĩa là quá trình ETL cần đủ mạnh để khi có lỗi gì thì chúng có thể recover mà không có sự mất mát hay hỏng dữ liệu nào

## ETL approaches and architecture

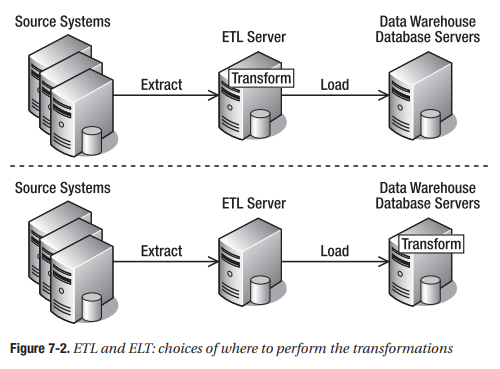
- Có một vài phương pháp tiếp cận trong việc cài đặt ETL. Phương pháp truyền thống đó là kéo data từ các source system, đẩy lên một staging area và sau đó biến đổi và cuối cùng là tải lên warehouse

- Bên cạnh đó, thay vì đẩy data lên một staging area nào đó thì đôi khi ETL server sẽ làm việc biến đổi ở trong bộ nhớ và update Data Warehouse trực tiếp luôn



- Biến đổi data trong bộ nhớ nhanh hơn việc đẩy chúng lên disk trước, đó là khi khối lượng data đủ nhỏ còn nếu không thì việc đẩy chúng lên disk là cần thiết

- Bên cạnh 2 hướng tiếp cận của ETL thì chúng ta còn có phương pháp ELT => Với hướng tiếp cận này, chúng ta vẫn sẽ kéo data từ source system, nhưng sau đó sẽ tải chúng lên Data Warehouse và Data Warehouse sẽ xử lý, biến đổi và update data đó. Mọi người thường sẽ sử dụng phương pháp ETL khi mà họ có một ETL server đủ mạnh và phần mềm đủ sức để có thể chạy các biến đổi và tiến trình data quality

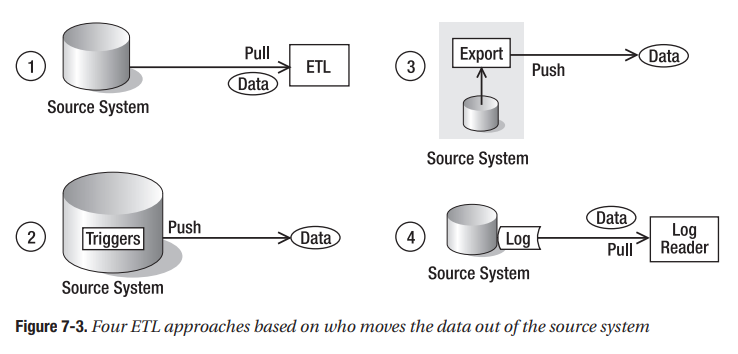


- Hệ thống csdl Data Warehouse mạnh ở đây thường sẽ là MPP database system, với các node giống nhau có memory, processor và disk cho riêng mình => nó cho hiệu suất cao hơn hệ thống chia sẻ ổ đĩa khi mà việc tải data diễn ra song song tới đa node

🡺 Ưu điẻm chính của MPP database system đó là hiệu suất tăng tuyến tính

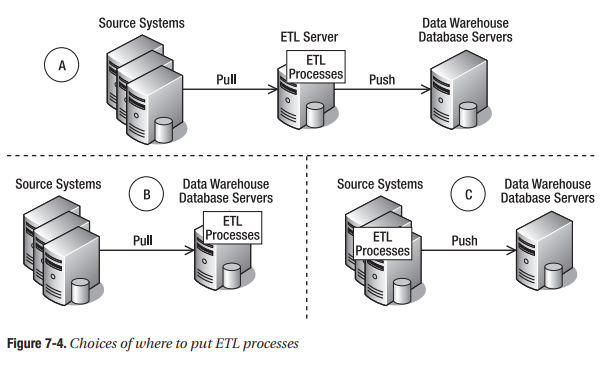
- In terms of WHO moves the data out of the source system => chúng ta phân loại các phương thức ETL thành 4 hướng tiếp cận:

1. Một quá trình ETL kéo data ra bằng cách truy vấn csdl source system một cách thường xuyên => Đây là hướng phổ biến nhất, ETL kết nối tới csdl source system, truy vấn dữ liệu và đem nó ra ngoài
2. *Triggers* trong *source system database* đẩy các thay đổi về data ra ngoài => Một database trigger là một tập hợp các dòng lệnh SQL khai thác ở mọi lúc, có thể là insert, update hay delete một bảng => Sử dụng triggers sẽ giúp lưu trữ các dòng có sự thay đổi ở một bảng khác
3. Một quá trình đã được lên lịch ở trong source system sẽ xuất data một cách thường xuyên => Có sự giống nhau với hướng 1, tuy nhiên chương trình dùng để truy vấn dữ liệu không phải *external ETL program* mà thay vào đó một *internal exporter program* sẽ chạy trong source system server
4. Một log reader đọc các database log files để xác định sự thay đổi của dữ liệu => Một file database log chứa các record của các transaction thực hiện với database đó. Một log reader là chương trình hiểu được format của data trong log file, chúng đọc, lấy dữ liệu ra và lưu trữ chúng ở một nơi nào đó



- In terms of where the processes that move the data out are executed => chúng ta phân loại ETL thành 3 hướng:

1. Khai thác quá trình ETL trong một ETL server riêng ở giữa source system và Data Warehouse server => Cho hiệu năng cao nhất, khi ETL có server riêng để chạy, nhưng nó cũng sẽ có chi phí bỏ ra cao hơn so với 2 hướng còn lại
2. Khai thác quá trình ETL ở trong Data Warehouse server => Hướng này có thể được sử dụng nếu chúng ta có capacity chưa dùng ở trong Data Warehouse server hay là có time slot khi mà Data Warehouse không sử dụng. Đương nhiên chi phí bỏ ra sẽ rẻ hơn khi không cần phải có thêm một server nữa
3. Khai thác quá trình ETL ở trong server mà host source system => Hướng này sẽ được cài đặt khi chúng ta cần một real-time Data Warehouse => Ngay khi có sự thay đổi về data trong source system thì chúng sẽ được lan truyền tới Data Warehouse, bằng cách sử dụng database triggers



## General considerations

- Hệ thống mà ta đang trích xuất có thể không phải là database, nó có thể là file system, queue, service hay email

- Nếu data ở trong database, thông thường sẽ lưu trữ data bằng ADO.NET, ODBC, OLEDB, JDBC hay các kết nối cơ sở dữ liệu độc quyền

- Nếu data ở trong một hay nhiều file, chúng có thể là structured, semistructured hay unstructured

- Một structure file chứa các dữ liệu bảng, có nghĩa là chúng thường ở dạng columns và rows, còn semistructured file chứa các tabular data và nontabular data, còn unstructured data sẽ cần phải được trích xuất thông text mining

- Việc trích xuất data có thể chỉ diễn ra trong các thời điểm cụ thể, hoặc không diễn ra trong các thời điểm cụ thể. Nếu may mắn thì có thể truy vập và dữ liệu nguồn bất kì lúc nào, ví dụ khi có một read-only server khác được cung cấp cho ad hoc queries và ETL

- Một trong những điều quan trọng nhất ở giai đoạn đầu của project đó là lấy được data, là việc có thể kết nối được tới source data. Thi thoảng sẽ mất thời gian để đến nơi mà chúng ta có thể truy vấn source system hay đọc files, khi mà source database server được đặt ở các châu lục khác nhau

## Extracting relational databases

- Sau khi kết nối tới dữ liệu nguồn, chúng ta đã có thể trích xuất được dữ liệu. Khi trích xuất data từ một csdl quan hệ gồm các bảng thì có thể dùng một trong 4 cách sau:

### Whole table every time

- Chúng ta sẽ dùng phương pháp này khi mà bảng của chúng ta đủ nhỏ, khoảng vài cột và vài dòng

- Với các bảng ít hơn 10000 dòng, chúng ta nên tính toán thời gian cần để tải whole table và so sánh với thời gian tải chúng theo hướng gia tăng với các ràng buộc

### Incremental extract

- Các transaction tables trong các tổ chức lớn sẽ là các bảng rất lớn với hàng triệu dòng nên nếu trích xuất cả bảng có thể phải mất cả ngày => Incremental extraction là kĩ thuật chỉ tải các dòng có thay đổi từ hệ thống

- Một vài thứ được dùng để trích xuất theo hướng gia tăng, đó là timestamp columns, identity columns, transaction dates, triggers, or a combination of them

### Fixed range

- Nếu như không khả thi để tải whole table vì nó rất lớn, và nếu như không thể tiến hành trích xuất theo hướng gia tăng khi không có cột timestamp nào tin tưởng,… thì chúng ta có thể dùng phương pháp “fixed range”

- Hiểu đơn giản, chúng ta sẽ trích xuất một số lượng cụ thể các bản ghi trong một thời điểm cụ thể

### Related tables

- Nếu một dòng ở trong bảng nguồn được update, chúng ta sẽ cần trích xuất dòng tương ứng ở trong related table

- Cái này cũng đúng với insert và delete, cả Data Warehouse application cũng vậy, nhưng lý tưởng nhất vẫn là được làm ở trong Data Warehouse database

- Để làm việc đó, chúng ta sẽ xác định dòng thay đổi ở trong bảng đầu tiên, sau đó dùng mối quan hệ của khoá ngoại và khoá chính để xác định dòng trong bảng thứ 2

### Testing data leaks

- Nếu chúng ta tiến hành trích xuất hướng gia tăng hay trích xuất theo khoảng, việc test cho data leaks là cần thiết.

- Khi đã trích xuất toàn bộ sự thay đổi trong source system đến Data Warehouse. Sau một khoảng thời gian cụ thể, ta tiến hành so sánh số lượng các dòng giữa source system và Data Warehouse, để xem xem có hay không một số dòng bị miss hay update bị miss bằng cách so sánh checksum

- Nếu chúng ta không có bất kì dòng hay update nào bị thiếu thì quá trình ETL gia tăng là có thể tin tưởng. Nếu có vài dòng bị mấy, hãy kiểm trả lại ETL logic, LSET, CET… Nếu mà ETL logic là chính xác thì hãy thay đổi hướng tiếp cận, có thể dùng trigger để capture các thay đổi của data hay sử dụng fixed-period extract

- Nếu chúng ta trích xuất từ relational database thì sẽ rất quan trọng khi luôn luôn test for data leaks. Nếu như mất dòng hay update thì có nghĩa là data trong Data Warehouse không thể tin tưởng => Bất kể chức năng Data Warehouse có tốt như nào, nếu data sai thì sẽ không khả dụng

## Extracting file systems

- Hầu hết các dạng của files mà hoạt động như sources trong quá trình ETL là các flat file

- 2 ví dụ về flat file đó là fixed-position files và pipe-delimited files

- Flat files phổ biến bởi chúng cung cấp hiệu năng tốt nhất. Importing hay exporting từ flat file là nhanh nhất, so sánh với các dạng file khác. Bên cạnh đó, SQL Server’s bulk insert (thêm hàng loạt) và bulk copy utility (bcp – tiện ích sao chép hàng loạt) làm việc cùng với flat files

- Bulk insert là một câu lệnh SQL load data từ một file lên SQL Server table, được thực thi từ bên trong SQL Server query editor

- Bulk copy utility được thực thi từ command prompt

- Một thực tế khá phổ biến trong kho dữ liệu cho hệ thống nguồn là xuất dữ liệu thành các flat file sau đó một quy trình ETL chọn chúng và nhập chúng vào kho dữ liệu. Dưới đây là một số điều bạn cần xem xét khi nhập từ flat files:

* Đồng ý với người quản trị hệ thống nguồn về cấu trúc của hệ thống file, bao gồm quy ước đặt tên file, cấu trúc thư mục
* Đảm bảo rằng chúng ta có thể truy cập vào địa chỉ cho phép, được phép truy cập để tiến hành thao tác với các file
* Đồng ý về quá trình
* Đồng ý về việc xử lý lỗi
* Đống ý về tần suất mà source system chạy export
* Đồng ý về format của file
* Đồng ý về cột nào được export bởi source system và theo thứ tự nào

- Trong SQL Server chúng ta sử dụng SSIS cho việc import các file Excel

- Web logs là các log files cho một web site, được định vị ở trong web servers. Mỗi web log là một text file chứa các yêu cầu về HTTP từ web browsers đến servers, gồm the client IP address, the date and time of the request, which page was requested, the HTTP code reflecting the status of the request, the number of bytes served, the user agent (such as the type of web browser or search engine crawlers), and the HTTP referrer (the page this request came from, that is, the previous page from which the link was followed)

- Mọi người rất là thích web logs bởi họ có thể gửi các thông tin hữu ích về browsing, shopping trong một web site thương mại điện tử. Mục đích về trích xuất từ web logs đến Data Warehouse để có thể tích hợp web traffic data với các data có sẵn trong warehouse

- Bên cạnh đó, database transaction log files và binary files được sử dụng như một nguồn của các quy trình ETL

- Database transaction log files được tận dụng bằng việc ứng dụng các transaction trong log files thành phiên bản copy thứ 2 của database, bằng cái sử dụng log shipping hay reading files và processing chúng sử dụng phần mềm chuyên biệt => rất hữu ích khi mà chúng ta không động đến primary database

- Transaction logs được xây dựng và back up cho mục đích recover csdl trong trường hợp xảy ra lỗi, bằng cách đọc và trích xuất các transactions tới Data Warehouse thì chúng ta tránh việc thay đổi source database

- Binary files chứa images, music samples, film trailers và documents có thể được import lên Data Warehouse bằng việc dùng tool như ADO.NET

- Ngày nay, các tệp XML ngày càng trở nên phổ biến hơn được sử dụng làm dữ liệu nguồn trong ETL. Chúng tôi có thể sử dụng SSIS để đọc, hợp nhất, xác thực, chuyển đổi và trích xuất tài liệu XML vào cơ sở dữ liệu SQL Server

## Extracting other source types

- Relational database và flat files là các dạng phổ biến nhất của dữ liệu nguồn cho các quá trình ETL của một hệ thống Data Warehouse, bên cạnh đó là spreadsheet files, web logs, hierarchical databases, binary files, database transactions logs và XML files

- Các dạng khác dữ liệu nguồn đó là web services, message queues và emails

- Một web service là tập hợp các chức năng biểu diễn các task chi tiết hay lấy các dữ liệu cụ thể, được hiển thị ra bởi web interface để ghi nhận yêu cầu từ web clients => Chúng ta không trích xuất web service mà dùng nó để lấy data => The benefit of using a web service to get the data is that the source system can have a single, uniform mechanism to publish its data. Tuy nhiên, web service chỉ nên được dùng để vận hành khi data đủ nhỏ, nếu data có khối lượng quá lớn thì không thể dùng web service được mà phải dùng một cơ chế khác cho việc tải data trước

- Một message queue là hệ thống cung cập giao thức kết nối đồng bộ, có nghĩa là người gửi sẽ không phải đợi cho đến khi người nhận lấy được thư

- E-mails được lưu trữ trong e-mail servers, được kết nối sử dụng các APIs, ADO.NET…

## Extracting data using SSIS

- Để có thể trích xuất data từ Jade (đối tượng trong case study - a custom Java-based system running on Informix) lên trên stage database thì chúng ta cần tiến hành theo các bước sau:

*1. Create the source system database and a user login.*

*2. Create a new SSIS project.*

*3. Create data sources for Jade and the stage.*

*4. Create a Data Flow task.*

*5. Probe the source system data.*

*6. Create a data flow source.*

*7. Create a data flow destination.*

*8. Map the source columns to the destination.*

*9. Execute the SSIS package*

- Khi chúng ta đối mặt với một source system chưa xác định, hãy cẩn thận vì chúng có thể là một bảng 500 triệu dòng hay chứa các kí tự lạ trong Extended Binary Coded Decimal Interchange Code (EBCDIC), cũng có thể ta sẽ quen với tên của các bảng và cột, với cú pháp của ngôn ngữ SQL trong nền tảng đó

## Memorizing the last extraction timestamp

- Việc ghi nhớ mốc thời gian trích xuất lần cuối sẽ giúp chúng ta biết được mốc thời gian gần nhất của data chúng ta đã trích xuất để ở lần trích xuất tiếp theo ta có thể bắt đầu từ mốc đó

- Bởi chúng ta đã sửa các giá trị tương ứng theo thời gian nên cần phải lưu trữ mốc thời gian trích xuất lần cuối ở trong metadata database để mọi lúc quá trình ETL chạy nó sẽ trích xuất các bản ghi khác nhau và sẽ chỉ trích xuất các bản ghi được thêm sau lần trích xuất cũ

- Trước khi bắt đầu, hãy xem qua tổng quan về những gì chúng ta sẽ làm. Trong vài trang tiếp theo, chúng tôi sẽ sử dụng dấu thời gian trích xuất cuối cùng để kiểm soát quá trình trích xuất tiếp theo. Điều này cho phép chúng tôi trích xuất bảng nguồn tăng dần. Để làm điều đó, trong vài trang tiếp theo, chúng tôi sẽ thực hiện các bước sau:

*1. Create a table in the metadata database to store the timestamps.*

*2. Create a data source for the metadata database.*

*3. Modify the data flow as follows:*

*a. Store the current time.*

*b. Get the last extraction timestamp from the table.*

*c. Add timestamps to the extraction query as parameters.*

*d. Update the timestamps in the table.*

*4. Clear the target table in the stage database.*

*5. Set the initial timestamps in the metadata table.*

*6. Execute the package to populate the target table.*

## Extracting from Files

- Với việc chúng ta đã trích xuất được data từ database thì bây giờ sẽ tiến hành trích xuất từ files

- Lấy ví dụ, ta có một ISO country data được cung cấp đến trong một file đa luồng, ta sẽ impory file này lên stage database

- Chúng ta sẽ tạo ra một SSIS package để import flat file lên stage database, sau đó sẽ lưu trữ file bằng việc di chuyển chúng đến một thư mục khác. Chúng ta sẽ hoàn thành chúng bằng việc tiến hành các bước sau:

*1. Download the flat file containing the ISO country data.*

*2. Open the flat file to see the content.*

*3. Create the target table in the stage database.*

*4. Create a new SSIS package.*

*5. Create an Execute SQL task to truncate the target table.*

*6. Create a Data Flow task to load the flat file to the stage database.*

*7. Create a File System task to archive the file.*

*8. Execute the package.*

- Mặc dù chúng tôi không trích xuất tăng dần ở đây, ta vẫn phải thêm metadata bit như đã làm trước đây khi trích xuất order header. Điều này có nghĩa là chúng ta cần lưu trữ dấu thời gian chạy thành công cuối cùng trong bảng data\_flow. Việc ghi lại kết quả của mỗi luồng dữ liệu ETL là rất quan trọng để khi ETL bị lỗi và ta muốn khởi động lại từ lỗi, ta biết bắt đầu lại từ đâu vì ta biết tác vụ nào thành công và tác vụ nào không chạy được. Ta sẽ thực hiện việc này bằng cách lưu trữ kết quả thực thi trong cùng một bảng siêu dữ liệu data\_flow

## Summary

- Chương này bắt đầu bằng các phương pháp tiếp cận và kiến trúc của hệ ETL, đó là ETL, ELT; sử dụng một server ETL đúng nghĩa hay không; push hay pull; lựa chọn server nào để đặt quá trình ETL

- Sau đó, chúng ta đã đi qua các dạng khác nhau của source system như relational database, spreadsheet, flat files, web logs, transaction logs, binary files, XML files images, web services, message queues và emails

- Ta cũng đã thảo luận làm thế nào để trích xuất hướng gia tăng một whole table mọi lúc, bằng cách sử dụng fixed-ranged và làm sao để detect data leaks

-Ta đã thảo luận về việc trích xuất sử dụng SSIS, làm thế nào được memorize the last extract để có thể trích xuất hướng gia tăng, bên cạnh đó là trích xuất từ file rồi đẩy chúng lên stage database bằng SSIS

# Populating the Data Warehouse

- Sau khi đã trích xuất data từ source system chúng ta sẽ điền thông tin vào NDS và DDS với data chúng ta đã có. Ở chương này sẽ có có 5 chủ đề chính về Data Warehouse population theo thứ tự chúng xảy ra trong một Data Warehouse system:

1. Loading the stage => focus ở đây sẽ là trích xuất data nhanh nhất có thể mà không cần làm quá nhiều transformation => focus on the loading
2. Creating the data firewall => Kiểm tra data quality khi data được tải từ stage lên NDS hay ODS => Việc kiểm tra sẽ hoàn thành khi sử dụng các rule đã định nghĩa từ trước để xác định hành động thực hiện: reject hoặc allow hoặc fix/correct data
3. Populating a NDS => bàn về chuẩn hoá data và key management
4. Populating dimension tables => Khi ta tải data từ NDS hay ODS lên bảng dim của DDS => bàn về những vấn đề như denormalization và SCD
5. Populating fact tables => Data từ NDS hay ODS hay stage (phụ thuộc vào data flow architecture ) được tải lên bảng fact => bàn về surrogate key lookup và late-arriving fact rows

- Chương này sẽ thảo luận và so sánh và batches, mini-batches và hướng tiếp cận near real-time, cùng với đó là real-time ETL => data sẽ được đẩy lên warehouse thay vì được kéo về

## Stage loading

- Stage có thể là một hệ thống các file hay một csdl.

- Khi sử dụng các files như là Data Warehouse stage thì cần lên một kết hoạch đặt tên file và cấu trúc thư mục cẩn thận, chúng ta cần phải truy cập đúng và được phép vào stage folders. Chúng ta cũng cần định nghĩa quá trình xoá và lưu trữ, xử lý lỗi và tần suất tải => QUAN TRỌNG NHẤT đó là định nghĩa cụ thể chi tiết về format của file

- Capacity trong ổ đĩa được phân bổ cũng là một nhân tố quan trọng của planning stage, phải đảm bảo rằng bạn sẽ không dùng quá dung lượng đang có. Disk space cho stage area được lên kế hoạch dựa vào khối lượng trích xuất daily data cực đại từ source system và bao lâu thì ta sẽ lưu trữ data

- Nếu stage là một csdl => KHÔNG nên đánh chỉ mục hay ràng buộc nào như not null, khoá chính… ở stage database => lý do chính là về hiệu năng bởi ta muốn capture và report những data xấu trong data quality process

- Đánh chỉ mục stage tables mô hình chung là không cần thiết, bởi cách mà stage tables đươc cấu trúc và lựa chọn => cách tốt nhất để làm việc đó là tải data lên stage tables rỗng không có đánh chỉ mục và chọn toàn bộ các record từ các bảng đó để tải lên NDS => Có 3 hướng về việc stage tables được cấu trúc như thế nào:

1. Keeping the previous day’s data in the same table
2. Keeping each day in a separate table
3. Having just one table and truncate the table every time before loading

- Vậy khi nào thì cần đánh chỉ mục cho stage table? Nó có thể cần thiết khi mà bạn sử dụng hướng tiếp cận 1 và stage table được sử dụng bởi nhiều úng dụng, không chỉ là Data Warehouse. Ngoài ra, nó còn phụ thuộc khối lượng data có đủ lớn hay không để có thể giúp ETL của ứng dụng chọn các dòng liên quan để một ngày cụ thể khi tải từ stage lên data store của ứng dụng

## Data firewall

- Data firewall rất quan trọng Data Warehousing bởi data firewall chặn các data xấu đi lên Data Warehouse

- Cái tên này được lấy cảm hứng từ chứng năng của một firewall trong network, các concept của một data firewall cũng giống với concept về firewall của network => data firewall là một chương trình kiểm tra incoming data (thường nó sẽ là một SSIS package hay quy trình được lưu trữ)

- Data firewall được đặt ở giữa stage và NDS, và việc reject hay cho phép data sẽ phụ thuộc vào data quality rules. Giống như tường lửa của mạng, data firewall có các cơ chế báo cáo rằng data gì đã bị loại bỏ, bị loại bỏ bởi rule gì và vào lúc nào => Mỗ khi data firewall capture hay tìm data xấu thì chúng sẽ được lưu lại ở data quality database cùng với rule được dùng để capture chúng, hành động gì được thực hiện và chúng xảy ra khi nào => báo cáo từ data quality database, setting data quality system để thông báo tới người phù hợp khi data quality rules bị vi phạm

- Không giống như tường lửa của mạng thì data firewall có thể fix và correct các bad data => khi mà data firewall xác định bad data, chúng ta có thể set chúng vào một trong ba hành động: reject the data, allow the data or fix/correct the data

- Data firewall là một phần quan trọng trong Data Warehouse loading bởi chúng đảm bảo cho data quality. Trước khi tải data lên NDS hay ODS hay DDS, chúng ta check data bằng việc đem chúng đi qua data firewall rules

## Populating NDS

- Điền vào một NDS sẽ tương đối khác so với việc điền vào stage => Chúng ta sẽ cần phải chuẩn hoá dữ liệu trước khi điền vào. Ta trích xuất data từ stage table hoặc thẳng từ source system và tải nó lên NDS database => Nếu record cho có thì insert, nếu có rồi thì update

- Sẽ có những vấn đề chúng ta cần phải xem xét như sau:

1. Normalization => ta cần phải chuẩn hoá data để nó fit với cấu trúc của NDS => cần phải điền các bảng cụ thể trước khi điền và bảng main
2. External data => khi ta tải data lên NDS thì có thể cần làm một số chuyển đổi dữ liệu
3. Key management => ta cần tạo và duy trì internal Data Warehouse keys mà sẽ được dùng trong DDS => Khi tải data lên NDS thì cần quản lý các key này
4. Junction tables => Cho phép ta cài đặt các mối quan hệ M-N và khi điền thì cần cần phải một trình tự cụ thể

### Normalization

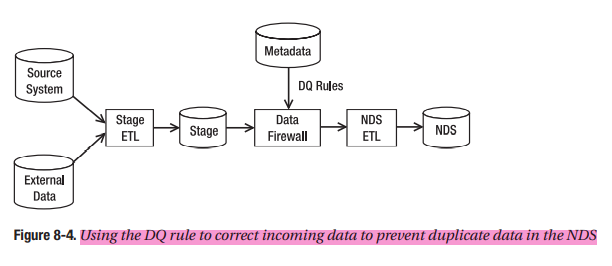
- Trong thực tế, khi bạn sử dụng một ODS hay NDS trong data warehouse architecture thì có rất nhiều các thực thể tham chiếu đã có sẵn trong source system, nhưng vẫn luôn có một vài thực thể không có ở trong source system và bạn cần tạo chúng dựa theo data của thực thể chính => Insert data từ các thực thể

### External data

- Chúng ta có thể đem data từ các nguồn bên ngoài khi điền vào NDS hay ODS

- Trong source system, ví dụ có data là một tên khác, trong khi đó external data có một tên khác => Khi lookup thì chúng ta sẽ không thể tìm ra => Chúng ta có thể áp dụng data quality routine để thay thế tên trong source system bằng tên mới của external data và đem chúng lên NDS

- Điều này rất cần thiết vì cần tránh việc có những đầu vào trùng lặp nhau cho ETL



### Key management

- Ta muốn cài đặt việc quản lý key khi đang điền vào NDS => mục đích của việc có cho riêng hệ thống key của data warehouse có 2 hướng:

* Nó cho phép sự tích hợp với source system thứ 2
* Ta muốn có thể thích ứng với sự thay đổi về khoá ở trong các hệ thống nguồn

- Data warehouse keying có form tăng dần từ 1 đến cuối cùng, data warehouse key giống như là surrogate key (SK), còn source system key giống như là natural key (NK). SK cho phép tích hợp giữa nhiều source system bởi chúng có thể ánh xạ hay liên kết NK vói SK

### Junction tables

## Using SSIS to populate NDS

## Upsert using SQL and Lookup

- Upsert có nghĩa là update nếu đã tồn tại, insert nếu chưa => Đó là một cách vận hành cơ bản trong data warehousing và bạn sẽ đi đến việc vận hành này rất nhiều lần khi xây dựng một hệ thống ETL cho data warehouse => Có 2 phương pháp khác thường dùng bên cạnh việc sử dụng SSIS để upsert đó là dùng câu lệnh SQL hoặc Lookup transformation

## Normalization

- Việc chuẩn hoá là key part của việc điền NDS

- Chúng ta sẽ thảo luận làm thế nào để tải stage data lên normalized NDS tables (lấy ví dụ về store table)

## Practical tips on SSIS

## Populating DDS dimension tables

- DDS là một dimensional store => các bảng ở trong DDS là không chuẩn hoá

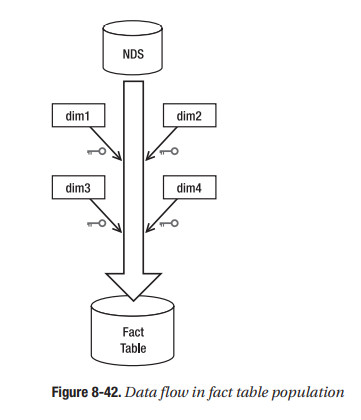
- Giống như tải data lên NDS thì khi tải data lên DDS ta cũng sẽ dùng upsert để update hoặc insert soucre row phụ thuộc vào nó đã tồn tại ở target hay chưa

- Ta có hai dạng chính của bảng DDS là bảng fact và bảng dim. Trước tiên là với bảng dim, khi tải data lên bảng dim trong DDS thì cần xem xét các vấn đề sau:

* Incremental loading => Khi tải stage data lên NDS, ta luôn đặt thời gian hiện tại vào cột last-updated nên sẽ biết thời điểm cập nhật lần cuối của từng record => Cho phép tải data của NDS lên DDS một các gia tăng, có nghĩa là chỉ cần tải các dòng của NDS đã thay đổi từ lần cuối ETL chạy
* Key management => Trong DDS ta không cần duy trì surrogate keys => Với NDS + DDS architecture thì ta có nhiều DDS. Vì ta định nghĩa và quản lý data warehouse surrogate key trong NDS nên phải đảm bảo dùng các DDS phải có cùng surrogate key =>> ODS thì khác, chúng vốn không có surrogate keys nên ta sẽ phải tạo và quản lý chúng ở trong DDS
* Denormalization => Để load một bảng dim đơn lẻ lên DDS thì ta cần phải lấy data từ một vài bảng ở trong NDS và nối chúng lại
* SCD => các bảng dim trong DDS có thể sẽ dùng SCD type 1, 2, 3 tuỳ vào mỗi bảng dim của các area khác nhau

## Populating DDS fact tables

- Cơ chế cơ bản trong việc điền bảng fact đó là đọc các source row từ NDS, look up đến dimensional keyss và insert chúng để bảng fact. Từ đây, việc điền vào bảng fact cần được hoàn thành sau khi bảng dim được điền xong



- Trong một số data warehouse thì ta chỉ dùng inserts trên bảng fact. Một số hệ thống khác thì sẽ dùng cả inserts và updates => Nếu ta cần update bảng fact thì sẽ dùng kĩ thuật upsert, ngoài ra ta vẫn có thể dùng câu lệnh SQL hay Lookup transformation

- Ta sẽ không thường xoá bảng fact, trừ khi phải xoá vĩnh viễn thì cancellations hay returns được cài đặt như inserts với chỉ số âm hay có định dạng khác

- Một hiện tượng khá thú vị khi ta điền bảng fact đó là late-arriving fact => khác với late-arriving dimension row thì chúng diễn ra khi ta nhận các dòng của bảng fact với thời điểm trong quá khứ (có thể là 1 tuần trước hay 1 tháng trước)

=> Chúng ta vẫn có thể làm như bình thường khi mà các transaction ở một vài tháng trước thì các dòng trong bảng dim có thể đã thay đổi. Nếu bảng kích thước đang sử dụng SCD loại 2, chúng ta sẽ có một số hàng trong bảng kích thước. Chúng ta cần tìm hàng nào tương ứng với giao dịch mà chúng ta tải vào bảng dữ kiện. Để làm điều này, chúng tôi sử dụng cột ngày có hiệu lực và ngày hết hạn trong bảng thứ nguyên. Hàng thứ nguyên chính xác cho giao dịch này là hàng có ngày giao dịch trong bảng dữ kiện nằm giữa ngày có hiệu lực và ngày hết hạn

## Batches, mini-batches and near real-time ETL

- Hầu hết các data warehouse ETL process chạy hàng ngày, hàng tuần hoặc hàng tháng => đây là batches

- Hiện nay chúng ta đã có mini-batch => một nhóm các quá trình ETL chạy theo giờ, thường là từ 1-6 giờ => chỉ MỘT VÀI quá trình ETL được bao gồm ở trong mini-batch, phần còn lại vẫn chạy trong các batches

- Do sự phát triển mới nhất này (hay nói cách khác là mini-batch), các kho dữ liệu giờ đây có một chức năng mới. Mười năm trước, kho dữ liệu được thiết kế cho mục đích phân tích chứ không phải là hệ thống hoạt động. Bây giờ họ có thể có một vai trò hoạt động

- Near real-time ETL là những điều người ta nói đến khi time lag giữa data trong data warehouse và data trong source system nhỏ hơn 1 tiếng, cụ thể là 10-30 phút. Và như mini-batches thì chỉ có một vài bảng được set trong fashion này

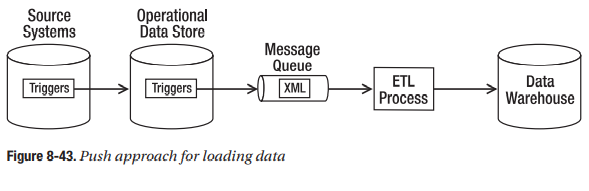
- Cũng giống như các quy trình theo batches, cả ETL theo mini-batch và gần thời gian thực đều dựa trên pull approaches. Các quy trình ETL kéo dữ liệu từ hệ thống nguồn vào kho dữ liệu. Sự khác biệt chính giữa ba trong số chúng là tần số chúng chạy. ETL hàng loạt chạy hàng ngày, hàng tuần hoặc hàng tháng; lô mini chạy hàng giờ; và ETL gần thời gian thực thường chạy cứ sau 10 đến 30 phút. ETL gần thời gian thực còn được gọi là micro-batch.

- Để áp dụng ETL theo micro-batch hoặc near real-time, bạn cần phân tích bảng dữ kiện DDS và xác định bảng nào sẽ có lợi nếu chúng được tải vài giờ một lần (hoặc ít hơn). Tốt nhất, bạn chỉ cần chọn một hoặc hai bảng dữ kiện để giảm thiểu tác động tiêu cực đến hiệu suất. Bước tiếp theo là xác định thứ nguyên nào liên quan đến các bảng dữ kiện này cần được tải ở cùng một tần suất. Sau đó, bạn có thể thay đổi khoảng thời gian thực hiện của quy trình ETL

## Pushing the data in

- Việc điền data warehouse đã được hoàn thành bằng việc sử dụng ETL => Quá trình ETL kéo data từ source system và tải chúng lên data warehouse

- Với các năm gần đây, vì chức năng vận hành mà một vài data warehouse cần phải update ngay khi source system thay đổi => source system sẽ push data lên data warehouse, và source system này sẽ có triggers, gồm insert and update triggers và cả update triggers như hình sau



- Khi ứng dụng chèn hoặc cập nhật hoặc xóa dữ liệu khỏi cơ sở dữ liệu hệ thống nguồn, các trình kích hoạt này được gọi và chúng chèn dữ liệu đã thay đổi vào ODS. Các bảng trong ODS cũng có các trình kích hoạt gửi các thay đổi dữ liệu đến hệ thống xếp hàng thư (MQ) ở dạng thông báo XML. Ở đầu bên kia của MQ, một ứng dụng ETL liên tục thăm dò và xử lý các thông điệp XML từ MQ, mở gói, cắt nhỏ chúng, gán hoặc tra cứu các khóa kho dữ liệu và chèn chúng vào các kích thước và bảng dữ liệu thích hợp trong kho dữ liệu. . Hoặc, loại bỏ MQ, trong trường hợp kiến trúc chỉ dành cho DDS, các trình kích hoạt trong hệ thống nguồn cập nhật DDS trực tiếp. Kiến trúc chỉ dành cho DDS phù hợp hơn với cách tiếp cận đẩy vì không có kho dữ liệu trung gian như NDS hoặc ODS. Thời gian dẫn đầu điển hình là từ 5 đến 30 giây, diễn ra trong thời gian thực.

- Ba năm trước, chúng tôi biết đến cách tiếp cận này chỉ được triển khai cho tích hợp ứng dụng doanh nghiệp (EAI). Ngày nay, nó được triển khai cho dân số kho dữ liệu. Một số người gọi cách tiếp cận đẩy này là tích hợp dữ liệu thời gian thực

- Một lưu ý quan trọng khi tải các bảng dữ kiện lớn trong thời gian thực là chúng ta cần sắp xếp ETL sao cho chỉ có thể chèn vào bảng dữ kiện. Chúng tôi không thể cập nhật hoặc xóa vì quá trình này sẽ mất vài giây và chúng tôi không có vài giây. Chúng ta chỉ có vài mili giây. Khi lô hàng ngày chạy, cùng với việc lật chế độ xem bảng dữ kiện, chúng tôi có thể xử lý dữ liệu được chèn của ngày hôm nay và cập nhật theo batches

## Summary

- Chương này bắt đầu với các thảo luận về stage loading, bao gồm 3 hướng tiếp cận để cấu trúc bảng stage

- Ta đã nói về data firewall, theo đó là một vài điểm cần xem xét khi tải lên NDS: normalization, external data, key management, and the many-to-many relationship

- Ta đã nói về 3 phương thức trong SSIS để tải data là lệnh SQL, SCD transformation và Lookup transformation

- Ta đã tiến hành điền bảng dim và bảng fact trong DDS, bao gồm SCD

- Cuối cùng ta đã thảo luận và batch, mini-batch và near real-time hay micro-batch như các hướng tiếp cận ETL, thực hiện bởi push approach

# Assuring Data Quality

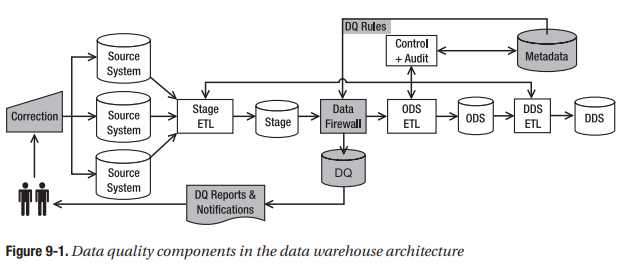
- Vì sao data quality lại quan trọng? Đây là một câu hỏi dễ dàng để trả lời. Nếu như bạn không tin tưởng vào data mình đang có thì có nghĩa là data đang có vấn đề, có thể về mặt chính xác (accuracy), có sự mất mát (completeness), sự bật đồng bộ (consistency), ước tính chỉ là thô (precision) hay dự liệu đã quá hạn để còn có thể sử dụng (timeliness)

- Xa hơn, nếu không thể tin tưởng data ở trong data warehouse thì chúng ta cần data warehouse để làm gì? Để tránh cho việc này thì khi xây dựng data warehouse, điều quan trọng đó là phải tìm hiểu về data quality càng sớm càng tốt và nên là ngay từ khi bắt đầu dự án

- Bản chất của việc duy trì data quality đó là tránh cho bad data có thể đi đến data warehouse và fix chúng sớm nhất có thể trong chuỗi quy trình hoạt động (thường sẽ là ở source system) => Để cài đặt như vậy, ta sẽ cần set up các rule để định nghĩa thế nào là bad data và thêm một số thành phần khác vào trong architecture để lọc được bad data ra và có thể kích hoạt cơ chế báo cáo, giám sát và làm sạch

## Data quality process

- Data quality process bao gồm các hoạt động đảm bảo cho data trong data warehouse là chính xác và hoàn chỉnh, đồng thời có các cơ chế để báo cáo bad data vào correct chúng => Sẽ có 3 khía cạnh của data quality process là checking; reporting và correcting



Các thành phần được bôi đậm ở hình trên chính là những thành phần có trong một data quality process của ODS + DDS data flow architecture, lần lượt là:

* Data firewall => chương trình kiểm tra incoming data, thường đó là một SSIS package hay một quy trình lưu trữ
* Metadata => csdl lưu trữ các data quality rules
* DATA QUALITY database lưu trữ bad data được phát hiện bởi data firewall
* DATA QUALITY reports and notifications đọc DATA QUALITY database và thông tin đến người có trách nhiệm cho data quality
* Correction là quá trình correct data ở trong source system

- Stage ETL trích xuất data từ source system và tải chúng lên stage database. Sau đó data firewall check data dựa theo data quality rules từ metadata database. Nếu data vượt qua được DATA QUALITY rule thì sẽ đi đến ODS ETL rồi sau được đẩy lên ODS, ngược lại nếu không vượt qua được thì sẽ được đưa đến DATA QUALITY database

- Khi mà data được đặt vào trong DATA QUALITY database, certain auditing information sẽ được ghi lại, bao gồm các thông tin như which source system the data is coming from, which table in the source system it is from, what time (and date) this happened, which rule(s) failed the data, and what table/mart/ area this data is for

- DATA QUALITY reports and notifications đọc DATA QUALITY database tại một khoảng thời gian thường nhật (such as daily). Chúng đọc các record ở trong DATA QUALITY database liên quan đến rule của chúng và báo cáo đến người thích hợp tại khoảng thời gian nói trên => Họ sẽ fix data ở trong source system để trong lần tiếp theo khi stage ETL trích xuất data nó sẽ hoàn toàn chính xác 🡺 Mục đích của reports and notifications là để thông báo users cần phải làm cho data được chính xác

- Có một nhóm báo cáo về data quality mà mục đích của nó đó là phân tích các data quality trong data warehouse, một trong các chỉ báo về quality đó là số lần mà từng rule bị vị phạm => Việc phân tích này thì có thể dùng data quality dashboards => đây là business intelligence applications mà đem đến một tông hợp nhanh về data quality dưới các biẻu đồ, chỉ báo, lược đồ…

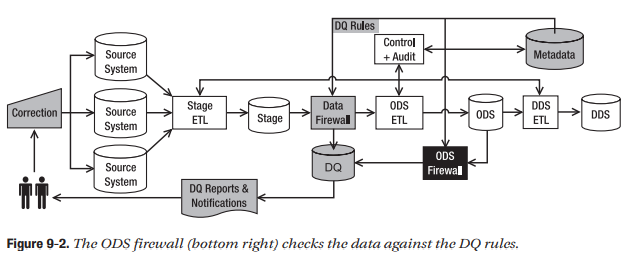
- Bên cạnh đó, chúng ta hoàn toàn có thể fix data một cách tự động ở trong data warehouse nhưng không nên bởi data ở trong source system vẫn sai => Nếu một ứng dụng khác sử dụng data đó thì sẽ publish data bị sai

- Vậy có thể được cho correction được tự động ở trong source system? Tất nhiên là có tồi, nhưng sẽ phải cẩn thận khi làm chúng. Thứ nhất đó là các luật cho việc automatic correction sẽ phải được check và double-check để đảm bảo rằng chúng 100% đúng. Thứ hai, nhìn chung sẽ tốt hơn nếu có thể hiểu được là bad data được tạo ra trong source system như thế nào và vì sao lại xảy ra điều đó => Nếu nó nằm trong tầm kiểm soát thì hãy fix chúng để không xảy ra thêm lần nào

- Trong thực tế, chúng ta sẽ có cách xử lý khác nhau cho các khoảng giá trị data khác nhau => có data đó sẽ được đánh dấu là warning, error…

- Với hình 9-1, chúng ta có thể đặt data firewall ngay trước stage ETL thì đổi lại quá trình trích xuất sẽ lâu hơn nhưng nó sẽ giúp chúng ta tránh khỏi bad data và chúng còn không thể có ở trong stage => Bên cạnh đó ta cũng có thể check data ở trong source system và không trích xuất bad data bằng việc sử dụng mệnh đề WHERE được set tự động => data từ source sẽ clean, nhưng việc trích xuất cũng sẽ chậm hơn. Nhược điểm sẽ là không thể báo cáo bad data một cách toàn diện bởi những data này không được trích xuất

- Vấn đề là nếu có một tập các bản ghi lớn thì làm thế nào để có thể báo cáo các sai phạm có trong đó => chúng ta sẽ tạo ra một ODS firewall => là một chương trình chạy sau khi data được tải lên ODS và trước khi được tải lên DDS



- Nếu cho ta cho phép data ở lại ODS thì cần đánh dấu data quality, bằng cách thêm 5 data quality keys trên mỗi dòng: accuracy, presicion, timeliness, completeness và consistency => tham chiếu đến 5 data quality dimensions

- Ngoài ra, ta có thể chỉ cần dùng một DATA QUALITY key duy nhất trên mỗi dòng => gọi là reliability và nó được tính toán dựa trên số điểm của 5 DATA QUALITY base dimensions => Nhược điểm là sẽ có tricky

## Data cleansing và matching

- Data cleansing hay data scurbbing là quá trình xác định và làm đúng diry data => incomplete, wrong, duplicate hay out-of-date data

- Trong data cleansing, việc có thể xác định một data item là giống với một thứ khác là rất quan trọng => đó là data matching, dùng để xác định các record trùng lặp khi thực hiện lookup với reference data => có liên quan đến character-based data types

- Trong SQL Server ta có ba dạng matching logic => exact, fuzzy (approximate) và run based:

* Exact matching => khi toàn bộ các kí tự đều giống nhau. Ví dụ: “Los Angeles” và “Los Angeles.” => dùng Lookup transfromation trong SSIS
* Fuzzy logic matching => tìm ra sự giống nhau giữa 2 set of data => dùng Fuzzy Lookup transformation trong SSIS
* Rule-based logic => dùng các luật cục thể data để xác định một việc nối

## Cross-checking with external sources

- Một số hoạt động của data cleansing được hoàn thành ở bên trong, một số khác thì được thực hiện bằng việc sử dụng tham chiếu chéo đến các data source bên ngoài

- Cross-referencing with external data có thể được thực hiện bằng việc sử dụng data từ các tổ chức bưu điện của từng quốc gia

- External data sources được vận chuyển với rất nhiều format và theo nhiều cách khác nhau => nó ảnh hưởng đến cách mà ta tích hợp các data bên ngoài vào trong data warehouse

- Ta cần phải làm việc với nhà cung cấp để có thể sử dụng data của họ hay các dịch vụ một cách hợp lý trong data warehouse. Xem xem external data được tích hợp vào data warehouse như thế nào, chúng ta có thể phân chia ra các cơ chế vận chuyển khác nhau thành 2 hướng tiếp cận lớn, đầu tiên là đem chúng đến data warehouse => provider đưa ta data và ta sẽ lưu trữ chúng trong data warehouse. Hướng thứ 2 đó là ta sẽ đưa data đến nhà provider và lấy chúng ra sử dụng khi cần, ví dụ là dùng web services

## Data quality rules

- Data quality rules là các bộ lọc một cần thiết để loại bỏ dirty data khỏi việc được đẩy lên warehouse. Dựa vào vị trí của data mà sẽ có 3 dạng data quality rules:

* Incoming data validation
* Cross-reference validation
* Data warehouse internal validation

- Incoming data validation là khi ta check data mà không có tham chiếu đến các data đã có sẵn trong warehouse như ODS, NDS hay DDS => được tiến hành nhanh chóng khi data được tải lên warehouse

- Cross-reference validation là khi ta check incoming data với data có trong warehouse => Mục đích là điểm đảm bảo rằng giá trị của incoming data trong khoảng cụ thể đã được tính toán dựa vào data đã có sẵn => được tiến hành nhanh chóng khi data được tải lên warehouse

- Data warehouse internal validation là nơi mà ta check data có sẵn trong warehouse => mục đích là xác minh chất lượng của data => Có thể hiểu đơn giản là có thể từng record sẽ đúng nhưng khi tổng hợp lại chưa chắc đã đúng => Thự hiện bằng cách so sánh tổng các record trong một khoảng thời gian cụ thể với một giá trị tiêu chuẩn đã biết trước => Không giống như 2 cái dạng trên thì data warehouse internal validation được thực hiện sau khi incoming data được tải lên warehouse hoàn toàn

- DQ rules for internal data warehouse validation có nhiều tần suất khác nhau tuỳ thuộc vào độ quan trọng của data và doanh nghiệp muốn được thông tin nhanh như nào khi có một sự cố xảy ra

- Ta sẽ cần phải cẩn thận với 2 dạng sau bởi chúng có thể ảnh hưởng đến hiệu năng của ETL => Ta sẽ cần tính toán các số liệu thống kê trước thì khi mà các rule chạy thì không tạo ra hiện tượng nút cổ chai và validation sẽ nhanh bởi ta đang so sánh một con số cố định thay vì một hàm toán

- Không phải tất cả DATA QUALITY rules là error, một số rule chỉ là warning hoặc informational only

- DATA QUALITY rules được mô tả trong một bảng là dq\_rules => mục đích để mô tả chi tiết từng data quality rule với các thông tin sau:

• The objective of the rule (including the calculation if any, in English)

• Whether it is an error, is a warning, or is informational

• What type of validation it is (whether it is an incoming data validation, cross-reference validation, or internal validation)

• How critical the rule is (measured by the risk), from level 1 (low risk) to level 5 (high risk)

• What actions to take, such as reject the data, allow the data into the warehouse, or fix it

• The status of the rule (whether it is active or disabled)

• Who to notify when there is a violation to this rule

## Action: reject, allow, fix

- Khi một data quality rule bị vi phạm có 3 lựa chọn để làm với data là có reject data, allow data lên data warehouse hoặc fix/correct data

- Reject action có nghĩa là ta không cho phép data lên warehouse mà sẽ đặt chúng vào một bảng audit trong DATA QUALITY database => được báo cáo đến business users và correct lại trước khi được tải lên data warehouse

- Thay vì reject data thì ta có thể allow chúng lên warehouse, bởi có thể chúng chỉ là một warning mà không phải error. Lý do thứ hai cho phép data lên warehouse là data khác trong cùng một dòng đó được yêu cầu cho việc tích hợp data

- Action khả dụng thứ ba đó là fix data dựa trên một số rule cụ thể, thường sẽ là tự động thay một giá trị khác phụ thuộc và tiêu chuẩn chi tiết => tuy nhiên nó cũng giống như ETL transformation thì chúng ta update data trước khi đẩy nó lên warehouse và data ở trong source vẫn sai => Sẽ luôn luôn tốt hơn khi có thể fix data trong source system thay vì fix chúng trong data warehouse

## Logging and auditing

- Khi một rule bị vi phạm thì chúng ta sẽ lưu trữ event đó ở trong data quality database. Chúng ta sẽ lưu trữ thời gian vi phạm, luật gì vi phạm, hành động nào được thực hiện và trạng thái correction => Ta lưu toàn bộ thông tin trên trong một bảng gọi là data quality log => quá trình mà ghi lại các sự kiện vi phạm luật cùng với các dòng đó là data quality logging

- Các dòng bị lỗi được lưu trong DATA QUALITY database chứ không phải DATA QUALITY log table, ở trong các bảng là data quality audit tables tương ứng với area trong doanh nghiệp mà chúng thuộc về

- Quá trình phân tích vi phạm data quality rules bằng truy vấn DQ log table và DQ audit tables là data quality audit => mục đích là để tìm ra when a DQ rule was violated, which/how many data rows are impacted, which/ how many source/target tables were impacted, what action was taken, what is the status, and what were the values of the source data. Một mục đích khác nữa đó là tìm hiểu về chất lượng của data trong data warehouse ( các luật gì bị vi phạm, số lần các luật bị vi phạm…)

- Dù action thực hiện là gì thì ta vẫn cần đặt các dòng lỗi vào bảng audit

=> Dùng cột lưu primary key để làm khác biệt so với target data warehouse table

- Các bảng audit và log cần được xoá vĩnh viễn hay lưu trữ sang một nơi khác có định kì để giữ cho số lượng dòng trong bảng main là nhỏ nhất. Tất nhiên, chỉ xoá những bản ghi đã được giải quyết

## Data quality reports and notifications

- Mục đích chính của DATA QUALITY reports and notifications đó là get somebody correcting the data

- Mục đích thứ hai đó là khiến cho users có nhận thức rằng một khu vực nào đó của warehouse có vấn đề về data quality

- Thông thường, khi một hệ thống DATA QUALITY được đẩy lên vận hành lần đầu thì số lượng vấn đề báo cáo sẽ lớn, sau đó nó sẽ giảm đều và có khi là theo cấp số nhân. NHƯNG phải nhớ rằng không bao giờ vấn đề phát sinh = 0 => Luôn có những vấn đề mới xảy ra và việc thay đổi DATA QUALITY rule thay vì correct data là tốt hơn

## Summary

- Trong chương này, ta đã thảo luận về data quality là gì và vì sao nó lại quan trọng

- Ta đã thảo luận về data quality process, data cleansing, data matching

- Ta đã thảo luận về các dạng của data quality rules (3 dạng) và 3 lựa chọn để làm với data khi data quality rules bị vi phạm

- Ta đã bàn về logging và auditing, cuối cùng là về reporting and notification để có thể có được người/nhóm sẽ correct data

🡺 Trong nhiều khía cạnh của data warehousing thì data quality có thể coi là thành phần quan trọng nhất, khi mà nếu data bị sai thì data warehouse tất nhiên sẽ không dùng được

# Metadata

- Hiểu chung chung thì metadata là data mô tả chính nó, toàn bộ các thông tin liên quan đến data

- Chương này sẽ thảo luận về metadata là gì trong ngữ cảnh của data warehousing, vì sao nó lại quan trọng? Nó được dùng để làm gì? Làm thế nào để lưu trữ chúng? Làm thế nào để duy trì chúng?

## Metadata in data warehousing

- Trong data warehouse thì metadata chứa các định nghĩa về data (the meaning and source of each column), the definition of the data warehouse itself (in other words, the data store structure, the ETL processes, and the data quality), the definition of the related systems (for example, the source systems), the audit information (what processes ran and when they ran), and the usage (which reports and cubes are used by whom and when) => Có 7 dạng metadata sau:

* Data definition and mapping metadata => chứa các định nghĩa về từng cột dim và fact và data đến từ đâu
* Data structure metadata mô tả cấu trúc của các bảng ở từng data store
* Source system metadata mô tả cấu trúc dữ liệu của source system database
* ETL process metadata mô tả từng data flow trong các quy trình ETL
* Data quality metadata mô tả data quality rules, các mức độ rủi ro và hành động của chúng
* Audit metadata chứa các bản ghi của các quá trình và hoạt động trong data warehouse
* Usage metadata chứa các nhật kí sự kiện của application usage

- Sao chúng ta lại cần metadata? Lý do chính và cơ bản đó là để mô tả và giải thích về data và về data warehouse (cấu trúc và các tiến trình) tới người dùng. Lý do thứ hai đó là cho mục đích kiểm tra để có thể hiểu những gì đã xảy ra, xảy ra ở đâu và khi nào

- Metadata được sử dụng bởi hệ thống để chạy các task vận hành, như là xác định hành động cần được tiến hành khi bad data được phát hiện bởi data quality rule và thứ tự các task của ETL cần được tiến hành

- Về việc duy trì (maintainability), ta sẽ dùng normalized forms trong metadata database => Data definition, data mapping, ETL metadata và audit metadata tham chiếu tới data structure metadata bằng việc lưu trữ table keys và column keys

## Data definition and mapping metadata

- Data definition metadata là một danh sách tất cả các cột từ mọi bảng trong DDS, ODS và NDS cùng với các ý nghĩa của chúng và giá trị ví dụ

- Mapping metadata mô tả từng phần của data đến từ đâu trong source system => có thể coi là data lineage metadata

=> Lợi ích của việc để mapping metadata và data definition metadata trong cùng một bảng đó là cấu trúc sẽ đơn giản hơn, còn nếu để chúng ở hai bảng khác nhau thì cấu trúc sẽ linh hoạt hơn và có ít sự dư thừa về dữ liệu

- Nếu ta muốn mô tả toàn bộ dòng dõi (lineage) của data => cần phải để data mapping metadata ở một bảng khác riêng

## Data structure metadata

- Data structure metadata chứa các bảng và cột ở toàn bộ các bảng và cột trong toàn bộ data store trong data warehouse cùng với data type của chúng

- Chứa table definitions, indexes, referential integrity (primary keys, foreign keys), constraints (null, unique), identity details, partition details, and views

- Data structure metadata khi kết hợp với data definition metadata tạo thành data dictionary

- Mục đích chính của việc tạo ra data structure metadata là để chỉ ra và giải thích cấu trúc của data warehouse tới người dùng => cần phải mô tả ít nhất là về data store, các bảng và các cột

- Mục đích thứ hai đó là một sự tham chiếu đến các metadata khác => Data structure metadata chứa tên, data type và mô tả của toàn bộ các bảng và cột trong các data store nên các metadata khác cần tham chiếu đến data structure metadata để lấy được các thông tin này

- Việc điền một cách thủ công là điền bằng tay với các mô tả nghiệp vụ là cần thiết để có được những mô tả có đầy đủ ý nghĩa và hữu ích

## Source system metadata

- Source system metadata khá là giống với data structure metadata, nó chứa cấu trúc dữ liệu của source system. Giống như structure metadata thì chí ít nó cần chứa database level, table level và column level => bởi data mapping metadata chỉ chứa duy nhất column key

- Source system metadata cũng chứa các thông tin khác như tần suất làm tươi của các bảng cụ thể => việc refresh rất hữu ích để tối ưu cho quá trình ETL, ngoài ra còn có data profile

- Mục đích chính của việc tạo source system metadata đó là chỉ ra và giải thích về cấu trúc của source system trong data warehouse tới người dùng

- Mục đích thứ hai đó là phục vụ như một sự tham chiếu cho các metadata khác

- Như đã nói ở trên thì việc đặt source system metadata cùng một nơi với data structure metadata hơn sẽ là tốt nhất thay vì là tạo riêng chúng ở các bảng khác nhau => Chúng có cấu trúc giống nhau => dễ dàng truy vấn khi không cần có điều kiện để nối bảng

- Làm thế nào để điền vào source system metadata? Không như data structure metadata khi hầu như việc đó được thực hiện tự động (trừ việc ghi các mô tả) thì trong hầu hết trường hợp ta không được phép truy cập vào để điền tự động cho source system metadata => Ta cần được cấp quyền quản trị để có thể truy cập vào source system database => Ta cần phải điền thủ công, và thường sẽ lấy các thông tin mà ta cần điền vào source system metadata khi tiến hành phân tích source system

- Có một thứ khác mà source system metadata có trong khi data structure metadata không có đó là source data profile => nó chứa các thống kê và thông tin về dung lượng mô tả về source data như là giá trị nhỏ nhất, lớn nhất, trung bình hay số lượng các giá trị trong khoảng của từng cột, số dòng, size của dòng và số cột của từng bảng 🡺 Source data profile rất có ích khi tạo các quá trình ETL để trích xuất data từ source system và khi ta cần biến đổi các quá trình ETL sau khi data warehouse đã được triển khai

## ETL process metadata

- ETL process metadata chứa bảng data flow, package, và status

- Mục đích chính của việc có ETL metadata đó là quản lý các ETL package trong SSIS, mục đích thứ hai đó mô tả các quá trình ETL

### Data flow table

- Mô tả tên và mô tả của từng SSIS data flow, what table they extract from, the destination table, what transformations are applied (only a descriptive text, not the exact formula), the parent package, their current status, the last successful extraction time (LSET), and the current extraction time (CET)

=> Những thông tin này được sử dụng bởi các quá trình ETL để xác định bản ghi nào trích xuất từ source system

### Package table

- Chứa tên và mô tả của từng SSIS package, lịch trình dưới dạng text, trạng thái và lần cuối chúng chạy

### Status table

- Chứa status code và description

- Để có thể restart một ETL batch từ điểm phát hiện lỗi, ta cần biết chính xác nơi nó gặp lỗi, cái gì được khai thác và không

## Data quality metadata

- Data quality metadata chứa data quality rules đã mô tả ở chương trước, bao gồm: rule name and description, rule type (for example, error or warning), rule category (for example, incoming data validation, cross reference validation, or internal data warehouse validation; see Chapter 9), risk level (for example, business or financial risk if this rule is violated, on the scale of 1 to 5), status (active or not), action (reject, allow, or fix; see Chapter 9), notification, create timestamp, and update timestamp

- Cột notification chứa link tới DATA QUALITY notification table mà chứa các thông tin về người cần được thông báo và làm thế nào để thông báo tới họ

- Ta có thể có thêm các đặc tính khác như là thông báo bao nhiêu lần, trong khoảng thời gian nào, trạng thái xác nhận (acknowledgement status)…

- Ta có thể có thêm bảng thứ 3 chứa thông tin chi tiết về người dùng

- Ngoài ra, bảng thứ 4 về mô tả nhóm người dùng khi ta có nhiều người dùng và muốn khai báo data quality tasks tới một nhóm người thay vì chỉ một cá nhân

## Audit metadata

- Audit metadata chứa các kết quả của mọi quá trình và hoạt động trong data warehouse, bao gồm: data loading process (ETL), creation of specific purpose DDS (mart), manual data modification (for example, updating DQ user table), DQ rule validation, security log/breach (for example, web portal login), system enhancements and modifications (for example, patches, schema changes and upgrades), data purging and archiving, database tuning, and maintenance (indexing and partitioning activities)

- Audit metadata được sử dụng cho việc quản trị data warehouse, bao gồm giám sát quá trình ETL, giám sát data quality, nâng cao các yêu cầu, tải các bản vá và giám sát việc truy cập bảo mật

- Ứng dụng thứ hai của audit metadata đó là cho việc xử lý lỗi, khi có một vấn đề hay mối nguy hại nào, ta có thể truy vấn audit metadata để hiểu được rằng chuyện gì đã xảy ra trong vòng vài tiếng trước

## Usage metadata

- Usage metadata giống với audit metadata trong việc nó chứa nhật kí sự kiện, nhưng sự kiện này là việc sử dụng của Reporing Services reports, Analysis Services cubes và data mining model

## Maintaining metadata

- Việc giữ cho metadata luôn được cập nhật rất quan trọng khi nó có thể dẫn điến sự hiểu sai của data warehouse và system có thể đưa ra các hành động sai nếu metadata sai

- Tiến hành phân tích tác động của yêu cầu thay đổi dựa trên siêu dữ liệu ánh xạ dữ liệu lỗi thời có thể dẫn đến các vấn đề về ETL, chẳng hạn như do chúng tôi hiểu sai vị trí của một phần dữ liệu hoặc do loại dữ liệu không chính xác

- Siêu dữ liệu định nghĩa và ánh xạ dữ liệu thường được xác định và tạo ở giai đoạn đầu của dự án kho dữ liệu. Sau khi được tạo, chúng sẽ được duy trì thông qua các thủ tục yêu cầu thay đổi. Thay vì thay đổi trực tiếp trên bảng siêu dữ liệu, thay đổi cần được thực hiện bằng cách sử dụng tập lệnh thay đổi dữ liệu. Tập lệnh thay đổi dữ liệu là một tập lệnh SQL Giao dịch có chứa phần khôi phục, chi tiết yêu cầu thay đổi (đi tới bảng nhật ký sự kiện) và cuộn về phía trước

- Quản trị viên DW cần chuẩn bị một tập lệnh để thay đổi siêu dữ liệu và gửi tập lệnh này để được thực thi theo một quy trình yêu cầu thay đổi. Tập lệnh chứa ba phần: nhật ký, khôi phục và triển khai

• The log section records the change request details into the event log table. The reason for doing this is for traceability and troubleshooting if something goes wrong.

• The rollback section tries to “undo” the change. This is necessary to prove that we can revert the changes.

• The implementation section applies the change to the metadata database

- Một số siêu dữ liệu cấu trúc dữ liệu có thể được duy trì tự động bằng cách sử dụng các dạng xem danh mục đối tượng của SQL Server. Truy vấn lấy dữ liệu từ các dạng xem danh mục đối tượng và chèn hoặc cập nhật các bảng siêu dữ liệu thích hợp. Ví dụ, chúng ta có thể sử dụng truy vấn được hiển thị trước đó trong chương này. Một số siêu dữ liệu cấu trúc dữ liệu vẫn cần được duy trì theo cách thủ công, chẳng hạn như mô tả nghiệp vụ đối tượng, phân cấp chiều và chi tiết SCD. Siêu dữ liệu hệ thống nguồn thường được tạo và điền trong giai đoạn phân tích hệ thống nguồn và sau đó được duy trì trong giai đoạn phát triển bằng cách sử dụng tập lệnh thay đổi dữ liệu để cập nhật bảng siêu dữ liệu (được minh họa trước đó), theo quy trình yêu cầu thay đổi. Sau khi kho dữ liệu được phát hành, siêu dữ liệu của hệ thống nguồn được duy trì bằng cách sử dụng các tập lệnh thay đổi dữ liệu, được điều chỉnh bởi quy trình yêu cầu thay đổi

- Siêu dữ liệu quy trình ETL được tạo và phổ biến trong quá trình phát triển các gói SSIS. Chúng tôi cần cập nhật siêu dữ liệu quy trình ETL mỗi khi chúng tôi sửa đổi các gói SSIS. Tương tự, siêu dữ liệu chất lượng dữ liệu (bảng quy tắc DQ, v.v.) cần được cập nhật mỗi khi chúng tôi sửa đổi các gói SSIS. Lý tưởng nhất là chúng ta cần sử dụng tập lệnh thay đổi dữ liệu được chuẩn hóa để cập nhật các bảng siêu dữ liệu, thay vì sử dụng các câu lệnh SQL trực tiếp để cập nhật các bảng. Tất cả các cập nhật đối với siêu dữ liệu cần được ghi lại trên bảng nhật ký sự kiện siêu dữ liệu kiểm tra. Vì vậy, điều này cần được bao gồm trong các tập lệnh thay đổi dữ liệu

- Audit metadata và usage metadata được duy trì như data warehouse được sử dụng. Audit metadata được điền bởi các quá trình ETL, quá trình phát hành bản vá và quá trình bảo mật. Usage metadata được điền bởi quy trình lưu trữ hay cổng thông tin web

## Summary

- Chương này đã định nghĩa metadata là gì trong data warehouse và chúng chứa những thông gì

- Có 7 dạng metadata là data definition and mapping metadata, data structure metadata, source system metadata, ETL processes metadata, data quality metadata, audit metadata và usage metadata

- Metadata giúp ta hiểu về cấu trúc và định nghĩa của data trong data warehouse, giúp ta hiểu về chính data warehouse hơn

- Metadata được dùng cho mục đích giám sát, kiểm tra và xử lý sự cố

# Building Reports

## Data warehouse reports

- Một report là chương trình sẽ lấy data từ data warehouse và hiển thị, trình bày chúng tới users trên màn hình hoặc là văn bản giấy

- Data strores nào mà data warehouse reports thường sẽ lấy data? Data warehouse reports thường sẽ lấy data từ DDS, ODS, data quality database hay metadata database, và có thể là từ multidimensional database

## When to use reports and when not to use them

- Data warehouse reports được sử dụng để trình bày các data về doanh nghiệp tới users, đồng thời cũng được sử dụng cho mục đích quản trị warehouse. Chúng ta sẽ có một số các dạng report phổ biến hay gặp:

* Dimensional reports => mục đích chính là trình bày DDS data tới users trong một cross-tab format
* Data quality reports => báo cáo về các thống kê của sự kiện vi phạm data quality rules, thường sẽ truy vấn csdl metadata
* Audit reports => đọc các bảng ghi nhật kí sự kiện trong một csdl metadata để có thể lấy ra kết quả và trạng thái của các quá trình và hoạt động của data warehouse như là các quá trình ETL, tạo mart, nhật kí bảo vệ, nâng cao hệ thống, xoá data, duy trì csdl…
* Usage reports => đọc các bảng nhật kí usage để lấy thông các reports, cubes và mô hình mining nào được sử dụng, khi nào và bởi ai
* ODS reports => ODS hoạt động như một điểm tích hợp từ nhiều source system cũng như là data store cho một ứng dụng vận hành
* DDS single dimension reports => Một vài DDS reports chỉ cần một dimension độc lập cho một mục đích nghiệp vụ cụ thể nào đó thay vì join một bảng fact với các bảng dim
* DDS drill-across dimensional => Một số DDS reports đào sâu vào phân tích từ một bảng fact tới một bảng fact khác

- Phía trên là các ví dụ về ở đâu và lúc nào data warehouse reports được sử dụng, tuy nhiên không phải tất cả các trường hợp đều cần đến reports và không thích hợp cần đến chúng:

* Khi data trong data warehouse reports có thể đã cũ sau một 1 ngày hay 1 tuần, phụ thuộc vào tần suất mà ETL update data warehouse
* Một số trường hợp thì việc sử dụng các công cụ OLAP để explore data trong data warehouse mà cụ thể là DDS sẽ phù hợp hơn việc dùng reports

- Mặt khác, với 2 case trên thì bạn cũng cần phải lưu ý những điều sau:

* Với first point => đôi lúc ta cần báo cáo không chỉ từ một source system mà là từ nhiều source system khác nhau và trong trường hợp này việc sử dụng data warehouse reports là hợp lý hơn, khi mà chúng tích hợp một vài source system vận hành
* Với second point => những case cụ thể sau thì dùng reports được ưu tiên hơn:

• To use OLAP tools, the users need to understand how the data is structured and organized, while to use reports we don’t, so it is easier for the users.

• If we use OLAP tools, we need to populate and process the OLAP cubes every day to refresh the data in the cubes, but when using reports, we don’t need to populate and process. So, reports are simpler to maintain.

• OLAP tools are suitable to report from the DDS because the DDS is in dimensional format. It is easy and natural to create OLAP cubes from the DDS. Creating OLAP cubes from the ODS is not that easy, because the ODS is in normalized format, not in dimensional format. So if you need to report from the ODS, it is easier to use reporting tools than to use OLAP tools

- Việc tạo báo cáo sẽ diễn ra theo trình tự sau:

* Ta sẽ xây dựng query
* Ta sẽ định nghĩa report layout
* Ta sẽ thêm các tham số
* Cuối cùng, ta sẽ thêm một số thứ giúp ích cho báo cáo nhưng mà không có ở trong case study

## Report Wizard

- Việc build một query khá là đơn giản với case study của chúng ta

- Tuy nhiên, trong một dự án thực tế thì khi truy vấn một DDS thì ta sẽ phải làm việc và thực hiện công việc với nhiều bảng dim và nhiều bảng fact hay measures hơn. Dẫu vậy, nguyên tắc vẫn là thế => Just adhering it

## Report layout

- Với section này, ta sẽ làm hoàn hảo báo cáo hơn, nâng cao giao diện (appearance) của báo cáo, sửa đổi title báo cáo, format lại numbers, thêm tiêu đề cho các cột và adjust size cột

- Ta có thể chỉnh sửa layout theo rất nhiều cách, bao gồm việc thêm cột, thêm header và footer cho một page, thay đổi font, cấu hình, cài đặt về sắp xếp lề (alignment), cài đặt khoảng cách cũng như cụ thể hoá toàn bộ properties của text box

- Ta cũng có thể tạo một báo cáo trong tabular form thay vì matrix form => Ta có thể explore nhiều hơn về Reporting Services layout features và sử dụng chúng trong reports của mình

## Report parameters

- Report parameters là một đặc tính quant rọng trong data warehouse dimensional report

- Trong một mô hình dimensional thì core data được lưu trữ trong một bảng fact mà được nối tới các bảng dim để ta có thể lọc và nhóm lại bằng các thuộc tính chiều

- Ở trong section này, ta sẽ biết cách để thêm các điều kiện vào câu truy vấn, bằng cách sử dụng các tham số trong câu lệnh WHERE

- Lưu ý, một tên biến/tham số luôn luôn phải bắt đầu bằng @

## Grouping, sorting and filtering

- Ta đã biết sử dụng Report Wizard để tạo ra một matrix report, cách để xác định các truy vấn SQL, cách để xây dựng và chọn ra layout phù hợp, cách để sử dụng các tham số để thay đổi truy vấn

- Grouping là phân loại các dòng hay các cột thành một collection => có thể group data bằng fields hay expression => được dùng để cung cấp các một tập các logic của data ở trong một bảng hay ma trận để tính toán tổng, đếm hay các expression khác trong group header hay footer

- Sorting được sử dụng để sắp xếp một cột cụ thể theo ascending hay descending

- Filterin có nghĩa là giới hạn các dòng theo các tiêu chuẩn cụ thể, sử dụng tham số thì ta có thể lọc data theo run time

- Hầu hết các người phát triển SQL đều cho rằng SQL query formatting khá là phiền phức và gây nhiễu nếu chúng không có tiêu chuẩn hay style => style là một lý do khác mà vì sao người phát triển SQL không chọn Query Builder mà thay vào đó sử dụng plain-text Generic Query Designer

## Simplicity

## - Khi ta có nhiều báo cáo tổ hợp phức tập hay nhiều báo cáo giống nhau cùng thực hiện một công việc, chúng ta sẽ cần biết người dùng có hay không muốn sử dụng các công cụ OLAP

- Ngày từ đầu của việc xây dựng dự án, ta cần nắm rõ ưu và nhược điểm của reports

- Nếu bạn lấy các yêu cầu rất phức tạp về báo cáo thì sẽ đáng để tìm ra việc có hay không users cần để explore data

## Spreadsheets

- Hoàn toàn có thể hiển thị và trình bày cũng như báo cáo data trong data warehouse tới users bằng cách sử dụng bảng tính như Excel

- Spreadsheets là perfect compromise giữa SSRS reports (static) và OLAP tools (flexible)

## Multidimensional database reports

- Từ SSRS ta có thể truy cập relational data stores ở trong data warehouse như DDS database

- Tuy nhiên data warehouse có thể chứa multidimensional database hay cubes và từ SSRS ta cũng có thể truy cập multidimensional databases

## Deploying reports

- Sau khi đã build report thì ta cần publish chúng trong 1 web server để business users có thể truy cập vào chúng thông qua web browsers của họ => đây có thể hiểu là deploying reports

- Reports được triển khai tới một nơi gọi là Report Manager trên web server

- Để gia tăng khả năng mở rộng thì ta có thể triển khai report trên một web farm. Reporting Services cũng hỗ trợ report server database được tải trên SQK Server cluster => Clustering tăng khả năng khả dụng của reports

## Managing reports

- Sau khi triển khai reports tới Reporting Services servr, ta cần quản lý chúng. Trong thực tiễn ta sẽ cần quản lý ba khu vực:

* Security => authentication and authorization. Authentication là xác minh định dạng người dùng. Authorization là bao gồm việc cho phép users cụ thể truy cập vào đối tượng cụ thể và giữ những người còn lại không được vào
* Subscriptions => request để đưa một báo cáo cụ thể tới một người dùng cụ thể tại một thời điểm nào đó bằng một phương thức nhất định
* Execution => là về quá trình chạy của reports => ta có thể thi hành một báo cáo theo ba cách là on demand, on demand from cache hay from a snapshot

### Managing report security

- Trong một vài việc cài đặt data warehouse thì toàn bộ người dùng có thể truy cập vào reports nhưng cũng khá hiếm. Hầu hết thì ta chỉ có thể truy cập vào một số báo cáo nhất định hay ít nhất là có một số báo cáo mà người dùng không được phép truy cập

- Một cách tốt để cấu hình report security đó là tạo các nhóm người dùng và gán cho nhóm đó security roles trong Reporting Services => mục đích là dễ duy trì kiểm soát hơn là người dùng cá nhân

- Trong Reporting Services, ta có thể tạo folder và đặt một vài report vào trong folder đó, từ đây ta có thể sắp xếp các group theo folders

### Managing report subscriptions

- Một subscription là một request vận chuyển một báo cáo cụ thể tới một người dùng cụ thể, sử dụng một phương thức cụ thể tại một thời điểm cụ thể trong một format cụ thể

### Managing reporting execution

- Ta có thể thi hành một báo cáo theo ba hướng: on demand, on demand from the cache, hay from a snapshot

- On demand có nghĩa là khi ta chạy báo cáo, Reporting Services sẽ luôn lấy data từ database

- On demand from cache có nghĩa là khi ta chạy báo cáo, Reporting Services sẽ lấy data từ cache, không fresh từ database => Reporting Services vẫn chạy báo cáo nhưng sẽ không lấy data từ database

- From a snapshot có nghĩa là mọi khoảng thời gian cụ thể Reporting Services sẽ chạy báo cáo và lưu trữ output của bá cáo, bao gồm cả layout và data. Những output này là snapshot => Snapshots được lưu trữ trong report server database. Trong trường hợp này, Reporting Services không thi hành báo cáo khi mà users truy cập vào báo cáo => hiển thị snapshot đến screen hoặc mail

## Summary

- Reports là một trong hai cách/phương tiện để lấy data từ data warehouse => với key role này thì reports rất quen trọng trong data warehouse

- Reports có thể lấy data từ một relational database cũng như là từ multidimensional database

- Reports được dùng bởi business users cũng như người quản trị data warehouse trong form của data quality reports, audit reports và usage reports => Cho mục đích doanh nghiệp thì reports được dùng chủ yếu cho BI

# Multidimensional Database

## What a multidimensional database is

- Một multidimensional database là một form của csdl nơi mà data được lưu trữ trong các cells và vị trí của từng cell được định dạng bởi một số lượng các thứ bậc gọi là dimensions (chiều) => Mỗi cell đại diện cho một sự kiện kinh doanh và giá trị của các chiều thể hiện địa điểm và thời gian sự kiện đó diễn ra

- Structure lưu trữ các giá trị tổng hợp cũng như các giá trị base, thường là trong compress multidimensional array format hơn là trong bảng RDBMS. Giá trị tổng hợp là các tổng được tính toán trước của các giá trị base

- Với bảng 2 chiều hay ma trận 2 chiều thì mỗi cell chứa một hay nhiều giá trị ước tính hoặc none/empty. Một cell đại diện cho một business event. Giá trị của các chiều thể hiển địa điểm và thời gian sự kiện diễn ra

Với MDB 3 chiều thì sẽ giống như là các cube (khối) hypercube (nếu có nhiều hơn 3 chiều) => Sự kết hợp của 3 chiều sẽ trỏ tới một cell ở trong cube. Một cell đại diện cho một business event. Giá trị của các chiều thể hiển địa điểm và thời gian sự kiện diễn ra

- Multidimensional database thường được dùng cho BI, đặc biệt là cho OLAP và data mining (DM). Ưu điểm của việc sử dụng multidimensional database cho OLAP và DM thay vì dùng một relational database đó là ta sẽ dùng ít dung lượng ổ đĩa hơn và có hiệu năng tốt hơn. Mặt khác, nhược điểm khi sử dụng chúng đó là thời gian tiến hành được yêu cầu cho việc tải csdl và tính toán các giá trị tổng hợp => Bất cứ khi nào relational source update thì MDB cần phải được update hay reprocess. Nhược điểm thứ hai đó là khả năng mở rộng khi MDB không thể mở rộng quá nhiều

- RDBMS là hệ thống quản lý một relational database. Bên cạnh đó hệ thống quản lý và vận hành multidimensional databases gọi là multidimensional database system (MDBMS), có thể coi nó là OLAP servers và cube engines

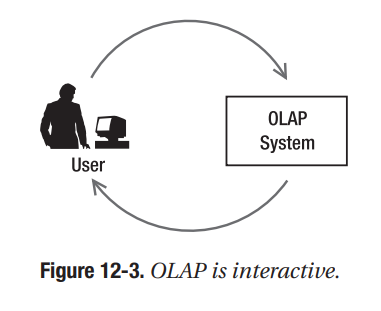
## Online analytical processing

- OLAP là hoạt động phân tích một cách tương tác business transaction data được lưu trữ trong dimensional data warehouse để có thể đưa ra các quyết định kinh doanh có chiến thuật và chiến lược

- OLAP giúp giải quyết các dạng vấn đề như là tính tổng các giá trị ước tính ở trong MDB với các tiêu chuẩn chi tiết phản ánh điều kiện doanh nghiệp. OLAP cũng cho phép business users đi đào xâu vào một area cụ thể của MDB để view data ở mức level chi tiết hơn

- Vậy ta có thể phân tích business transactions data được lưu trữ trong hệ thống vận hành như ERP systems hay là ODS? Đây là sự phân tích về nghiệp vụ kinh doanh nhưng không phải là OLAP

- OLAP có tính tương tác khi BA truy vấn thì hệ thống phản hồi, vòng lặp đó được lặp lại, với thời gian phản hồi nằm trong khoảng vài giây



- Hoạt động OLAP là một phần của BI. BI có 3 hoạt động chính đó là reporting, OLAP và data mining

- Chức năng OLAP có thể được chuyển đến thông qua relational database hoặc multidimensional database. OLAP dùng relational database là ROLAP và OLAP dùng multidimensional database là MOLAP. Ngoài ra còn có Hybrid online analytical processing (HOLAP) sử dụng cả 2 csdl trên

## Creating a multidimensional database

## Processing a multidimensional database

- Nếu ta có một dimensional relational data warehouse hay là DDS thì ta có thể dễ dàng tạo ra multidimensional database hay cube

- Trước khi ta có thể lướt cub hay truy vấn chúng cho việc báo cáo, bạn cần phải xử lý cube trước tiên. Processing a cube có nghĩa là tính toán các tổng hợp từ các cube cell và trước khi process chúng ta cần xây dựng (build) và triển khai (deploy) chúng đầu tiên:

* Building a cube có nghĩa là kiểm tra cấu trúc bên trong của cube, bao gồm facts, dimensions, hierachy và connections tới data source để đảm bảo cấu trúc ta tạo ra có tồn tại và có nghĩa
* Deploying a cube có nghĩa là tạo ra các cấu trúc cube mà ta đã thiết kế trong multidimensional database engine (a.k.a OLAP server or cube engine)

- Thông thường, một viễn cảnh thường thấy trong một dự án data warehouse đó là thêm hầu hết các dimensions và measures. Ta ưu tiên việc xây dựng các cube này từng chút một => giúp ta dễ dàng xử lý các vấn đề phát sinh

- Khi xây dựng một cube thì ta không cần phải thêm toàn bộ các dimensions và measures trong bảng fact

- Một số công cụ BI không yêu cầu các cube phải được build bởi họ có ROLAP hay reporting tools

- Ta không cần phải build một cube từ chỉ một bảng fact mà có thể từ 2 hoặc nhiều hơn các bảng fact và khi đó, ta cần đảm bảo rằng về dimensional conformity

## Querying a multidimensional database

- Bây giờ chúng ta đã có cube được xử lý và ta có thể truy vấn cube. Cách dễ nhất để truy vấn một cube bằng cách lưới chúng (browsing it). Browsing a cube có nghãi là truy vấn cube và hiển thị cube data trên màn hình trong một format và thứ tự cụ thể

## Administering a multidimensional database

- Một khi ta đã xây dựng, triển khai, tiến hành xử lý và truy vấn cubes thì ta cần quản trị chúng, thường sẽ có 3 common administrative tasks:

* To secure the cube
* To keep the data within the cubes up to date
* To make sure we can store the cubes if they are out of order

### Multidimensional database security

- Để bảo vệ cubes thì ta cần hiểu Analysis Services security architecture

- Ta có 2 role chính:

* Server role => được phép truy cập vào mọi thứ
* Database role => chỉ được phép truy cập vào một multidimensional database cụ thể và các cube ở trong chúng

### Processing cubes

### Backup and restore

- Tần suất backup sẽ được xác định phụ thuộc vào việc cube được xử lý thường xuyên không

- Một điều quan trọng mà ta cần phải nhớ đó là khi ta bạc up multidimensional database thì ta cũng cần restore backup không là ta sẽ gặp phải những rủi ro lớn

- Ta sẽ cần biết các concept quan trọng khi mà backup một MDB:

* Nếu MDB storage là ở trong ROLAP mode => backup của MDB chứa chỉ metadata và backup không chứa data
* Nếu MDB ở trong MOLAP mode => backup của MDB chứa data, aggregates và metadata
* Nếu MDB ở trong HOLAP mode => backup file chứa chỉ metadata và aggregations và không chứa base data

## Summary

- Ta đã tìm hiểu về MDBs, OLAP và sự khác biệt giữa chúng

- Ta đã thảo luận về sự khác nhau giữa ROLAP, MOLAP và HOLAP

- Ta đã tìm hiểu làm thế nào để process, query, secure và backup multidimensional database

# Using Data Warehouse for Business Intelligence

- BI là một tập các hoạt động để có thể “get an understanding and insights about a business” bằng việc tiến hành nhiều dạng phân tích trên company data cũng như external data từ các bên thứ ba để có thể giúp đưa ra “strategic, tatical and operational business decisions” và tiến hành đưa ra các hành động cần thiết để cải thiện hiệu suất, hiệu quả của doanh nghiệp

- Mục đích của bất kì BI activity nào đều là cải thiện business performance, một số ví dụ của BI implementations đó là phân tích khả năng sinh lời từ người dùng, nghiên cứu về khả năng sinh lời của sản phẩm, tính toán doanh số bán hàng đối với các sản phẩm và vùng miền khác nhau, khám phá khả năng sinh lời của accounts, đánh giá hiệu quả của nhà cung cấp và tìm hiểu các dạng rủi ro của người dùng

- It is important to bear in mind that business intelligence activities are not limited to the data in the data warehouse

- Ta có thể phân loại các ứng dụng BI thành 6 nhóm:

* Report applications => truy vấn vào data warehouse và hiển thị data trong static tabular hay pivot format, thường được dùng để tiến hành các *lightweight analysis*
* Analytic applications => truy vấn data warehouse có lặp lại và tương tác, sau đó hiển thị data trong các flexible format mà users có thể slice and dice, thường được dùng cho các *deeper analysis*
* Data mining applications => exploring data warehouse để tìm ra các pattern và các mối quan hệ mô tả về data
* Dashboards => là một dạng của ứng dụng BI mà đem đến a quick high-level summary của business performance dưới dạng *graphical gadgets, typically gauges, charts, indicators, and color-coded maps* => ta có thể đi sâu vào chi tiết các data ở level thấp hơn
* Alerts => notifications tới users khi có sự kiện cụ thể hay điều kiện nào đó xảy ra
* Portal => BI portal là một ứng dụng mà chức năng của nó như là một gateway để truy cập và quản lý BI reports, analytics, data mining, và ứng dụng dashboard cũng như là alert subscriptions

## Business intelligence reports

- Reports sẽ truy vấn vào data warehouse và hiển thị data dưới dạng tabular hay pivot format, tạo ra nhiều dạng chart khác, đồng thời ta có thể thêm các tham số vào các report để giúp nó trở nên sinh động

🡺 Ưu điểm chính của việc dùng reports trong BI đó là sự đơn giản => Reports rất đơn giản để tạo, để quản lý và để sử dụng. Ta sẽ dùng reports trong BI khi mà yêu cầu về format hiển thị tương đối đơn giản và không có sự thay đổi nhiều

🡺 Nhược điểm của việc sử dụng reports đó là chúng không có độ linh hoạt và khả năng tương tác

- Các báo cáo BI không chỉ giới hạn trong data warehouse mà chúng còn có thể truy vấn các nguồn data khác ở trong enterprise

=> Sử dụng reports thì ta có thể kết hợp các kết quả được lưu trữ của saved query và fresh output của online query

=> Ta có thể kết hợp data từ hệ thống ERP, ODS và data warehouse trong một single report hay một set of reports

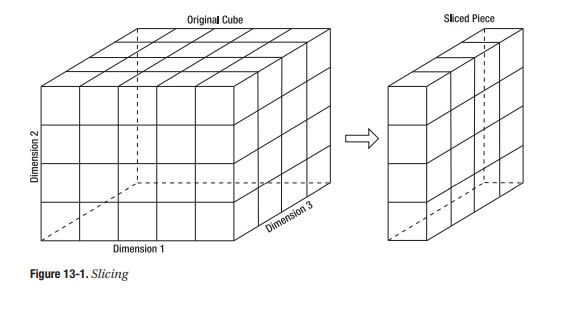
- Các báo cáo BI thường xuyên được quản lý và publish trong một địa điểm trung tâm, thường là *intranet*

## Business intelligence analytics

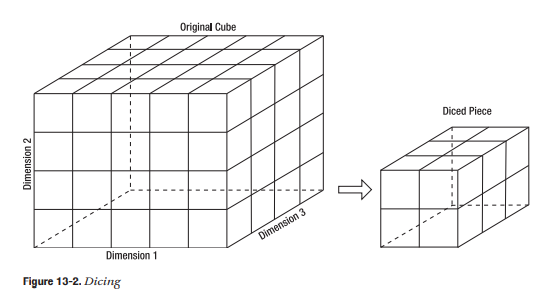
- Analytic applications được biết đến từ năm 1997 với định nghĩa chúng là những ứng dụng cung cấp sự đảm bảo về quá trình decision-making bằng việc truy cập time-based data từ nhiều nguồn

- Ta sẽ làm rõ những thuật ngữ mà chúng ta sẽ sử dụng trong việc phân tích, bao gồm slicing, dicing, drilling up và drilling down để có nhận thức giống nhau

=>> Slicing là quá trình lấy ra một block of data từ một cube bằng việc lọc theo một dimension

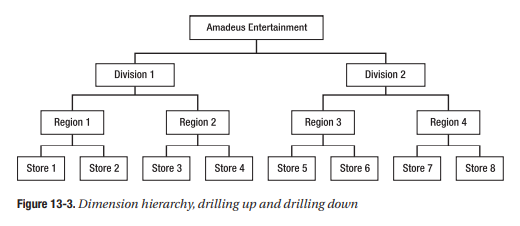


=>> Dicing là quá trình lấy ra một block of data từ một cube bằng việc lọc theo toàn bộ các dimension của cube đó



=>> Drilling up là việc hiển thị, trình bày data ở một level cao hơn trong dimension hierarchy (from detail level to summary level)

=>> Drilling down là việc hiển thị, trình bày data ở một level thấp hơn trong dimension hierarchy (from summary level to detail level)



- Analytics là những gì ta hay nghĩ đến nhất khi ta nói đến các ứng dụng BI

- Phụ thuộc vào ngành công nghiệp, dịch vụ thì ta sẽ sử dụng các ứng dụng khác nhau để đánh giá hiệu quả doanh nghiệp, phân tích khả năng sinh lời của sản phẩm và từ khách hàng và nghiên cứu việc làm thế nào để hạ chi phí lưu kho

- Một số ứng dụng phân tích có hệ thống quản lý csdl nhiều chiều riêng, một số thì dùng của các vendor => gọi chung là MOLAP applications => là các ứng dụng phân tích mà read from multidimensional databases

- Một số khác thì không như vậy mà read from relational databases => gọi chung là ROLAP applications

- Một số ứng dụng có thể read from both relational and multidimensional sources và combine data từ bên trong trước khi hiển thị chúng tới end users => Dù là gì thì data source luôn cần phải ở dimensional data warehouse form, bao gồm các bảng fact và dim that make comforted data marts

- Sử dụng các ứng dụng phân tích giúp ta các một cái nhìn tổng quan về hiệu suất hiện tại của doanh nghiệp, such as sales, production, purchasing, or profitability (probably by region) and compare it with the budget or targets

- We can also do charting, “what if?” analysis, planning exercises, and key performance indicator monitoring from analytic applications. We can swap any dimension with another dimension to get a different business perspective, going up and down the dimensional hierarchy to get different levels of summary, and we can pick certain dimension members and focus our attention on them. Using analytic applications, we can perform ad hoc queries, drill across to another cube, analyze data from several cubes, find and locate certain data, and perform various kinds of formatting and manipulation with regard to the results, such as adding or removing columns and performing calculations.

🡺 Ưu điểm chính của việc sử dụng analytic applications đó là độ linh hoạt (flexibility)

🡺 Nhược điểm chính của analytic applications đó là độ phức tạp (complexity) => Sẽ mất một thời gian cho users để có thể làm quen với các công cụ, đồng thời multidimensional databases data sources cần được duy trì và vận hành một cách hợp lý

=> Dẫu vậy, một khi mà mọi thứ đã được tổ chức và sẵn sàng hoạt động thì sẽ rất dễ dàng để để vận hành chúng

## Business intelligence data mining

- Data mining là một quá trình khai phá dữ liệu để tìm ra các pattern và relationship mô tả data và dự đoán về các giá trị tương lai và unknown của data

- Key value của data mining là khả năng hiểu được vì sao một số thứ diễn ra trong quá khứ và khả năng có thể dự đoán những gì sẽ xảy trong tương lai

- Đề cập đến việc dự đoán tương lai liên quan đến data mining thì ta có thể sử dụng thuật ngữ “forecasting” hay “predictive analytics”

- Nếu data mining được sử dụng để lý giải cho những tình huống hiện tại hay quá khứ thì đó gọi là descriptive modeling, descriptive analytics hay knowledge discovery

- Việc cài đặt data mining trong doanh nghiệp đang phát triển từng ngày, cả về descriptive và predictive analytics. Sử dụng data mining sẽ giúp tìm ra sự tương quan giữa purchase patterns và customer demographics

- Mô hình chung, sẽ có 4 bước để tiến hành việc cài đặt data mining:

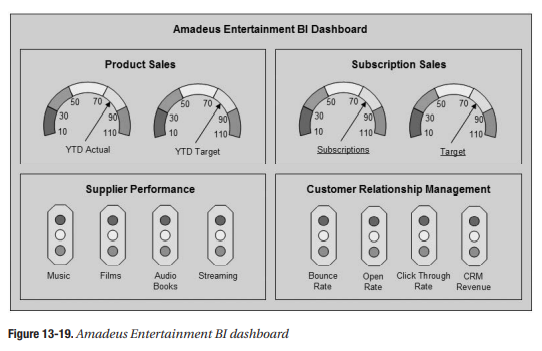
1. Define what we want to achieve => Ta sẽ lên các câu hỏi mà ta cố gắng phải trả lời
2. Prepare the data => Ta đảm bảo rằng ta có relevant data và kiểm trả về data quality
3. Build the mining models => Ta sẽ tạo một cấu trúc khai phá bao gồm một hay nhiều mô hình data mining. Mô hình khai phá có thể sẽ sử dụng các giải thuật khai phá khác nhau. Chúng có thể được xây dựng từ relational sources hay từ OLAP cubes. Ta sẽ xử lý các mô hình và kiểm thử cách mà chúng tiến hành chạy
4. Deploy and maintain the models in production => Ta có thể sử dụng các mô hình để tạo các prediction, dù là sử dụng SQL Server data mining language hay Prediction Query Builder. Prediction là sự dự đoán hay dự báo về các giá trị một biến cụ thể. Ta cũng có thể tạo ra các báo cáo truy vấn các mô hình khai phá bằng việc tạo ra một Analysis Services data set. Ta có thể cần xử lý các mô hình thường xuyên để giữ cho thông tin luôn được cập nhật

## Business intelligence dashboard

- Dashboards là một danh mục của business intelligence applications mà đem đến một cái nhìn tóm gọn ở level cao của business performance trong graphical gadgets, typically gauges, charts, indicators và color-coded maps

- Các ứng dụng Dashboard dựa vào data warehouse sẽ được update với tần suất phụ thuộc vào tần suất mà business perfomance measurement yêu cầu

- Real-time dashboards đang ngày càng trở nên phổ biến, truy vấn data từ hệ thống vận hành hơn là từ data warehouse



- Các ứng dụng Dashboard được sử dụng đầu tiên bởi upper-level management (có thể là quản lý cấp cao và uỷ viên ban quản trị mức C-level) để phân tích hiệu quả doanh nghiệp

- Today all levels of management use dashboards as the applications become increasingly closer to operational level rather than for analysis

- Các ứng dụng Dashboard đang trở thành các công cụ quan trọng và hữu ích cho việc quản lý việc vận hành của doanh nghiệp và “chèo lái” hiệu quả của doanh nghiệp

- Các chỉ số hoạt động quan trọng là sự đo lường phản chiếu các nhiệm vụ của doanh nghiệp => Các doanh nghiệp khác nhau sẽ có các KPI khác nhau

- Các ứng dụng Dashboard kết hợp các KPI trở thành các tiện ích trực quan mà rất dễ để hiểu và được cập nhật thường xuyên

=> Có thể cho biết các xu hướng hay tiến triển (lên hay xuống) của KPI bằng việc so sánh giá trị hiện tại và tới các giá trị trong quá khứ

🡺 Ưu điểm chính của các ứng dụng dashboard là khả năng hiểu được trạng thái tổng quan của toàn bộ doanh nghiệp trong một quick glance (một cái nhìn lướt nhanh qua) và khả năng có thể đi sâu vào thông tin dữ liệu ở một mức độ chi tiết nếu cần

🡺 Nhược điểm là việc để xây dựng một ứng dụng như vậy là không hề dễ và phải được cập nhật một cách thường xuyên

=>> Reports là ứng dụng dễ dàng nhất để build, kế đến là OLAP analytics và sau đó là data mining => Cái độ phức tạp ở đây là bởi vì ta cần có khả năng đào sâu vào chi tiết ở level thấp hơn và bởi vì đặc điểm hành vi tương tác của các đối tượng trực quan

## Business intelligence alerts

- Alerts là các thông báo tới users về một sự kiện cụ thể hay điều kiện đã diễn ra

- Ta có thể phân biệt các BI arlet thành hai dạng nếu chiếu theo hệ thống mà họ lấy data từ đâu, đó là:

* Data warehouse alerts
* Operational system alerts

=> Sự khác nhau chủ yếu giữa hai alert này đó là phạm vi (scope) của operational system alerts đó là mỗi transaction, trong khi data warehouse alerts là ở aggregate level

🡺 Mục đích chính của việc có BI alerts là có thể chạm tới các vấn đề một cách nhanh chóng để có thể làm giảm việc xấu có thể xảy ra hay chỉnh sửa một số điều trước khi nó trở nên tệ hơn

🡺 Ưu điểm chính của việc vận chuyển thông tin BI sử dụng alerts nó khác hoàn toàn với 4 phương thức trên đó là về timing

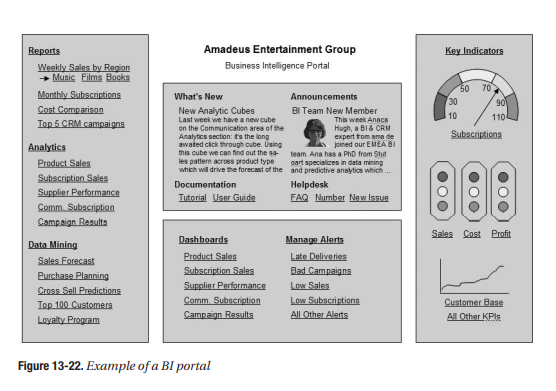
🡺 Nhược điểm chính đó là việc alerts thường xuyên chứa các data đơn giản, ngược lại với rich data mà được vận chuyển bởi data mining hay các ứng dụng dashboard => tuy nhiên alerts có thể chứa đường dẫn để web page mà chứa rich information

=>> Làm cách nào để vận chuyển alerts trong SQL Server? => Sử dụng SQL Server Notification Services. Notification Services là một nền tảng cho việc phát triển và triển khai các ứng dụng mà sinh ra và gửi thông báo đến người đăng kí => Có bốn thành phần:

* Subscription Management => lưu trữ người dùng đăng kí, thông tin được vận chuyển và các kênh vận chuyển được ưu tiên
* Event Collection => thu thập các sự kiện từ databases, hệ thống file hay các ứng dụng và gửi những sự kiện này đến ứng dụng thông báo
* Subscription Processing => đánh giá các subscription để tìm ra các match giữa các sự kiện được thu thập và từ đó sinh ra các thông báo
* Notification Distribution => chuyển dạng các data về thông báo sang các tin nhắn có thể đọc được, format lại tin nhắn và gửi chúng đến các người dùng đăng kí thông qua kênh vận chuyển được xác định từ trước

## Business intelligence portal

- BI portal là một ứng dụng mà chức năng của chúng như là một cổng (gateway) để truy cập và quản lý các báo cáo business intelligence, analytics, data mining, các ứng dụng dashboard và alert subscriptions



- Chúng có thể được phát triển như một ứng dụng được build theo mong muốn, hoặc có thể sử dụng giải pháp bên ngoài của các vendor

🡺 Mục đích chính của việc sử dụng một BI portal đó là toàn bộ các ứng dụng BI đều sẽ khả dụng trong một địa điểm trung tâm. Mục đích thứ hai đó là việc quản lý về bảo vệ được tập trung => Người dùng chỉ cần đăng nhập một lần vào cổng thông tin và sẽ không cần phải đăng nhập lại để truy cập vào từng ứng dụng nên sẽ dễ dàng cho quản trị viên có thể quản lý về security

- Ta có thể để thêm các tài liệu BI, hướng dẫn người dùng cũng như là sđt hỗ trợ và đường link tới bộ phận giúp đỡ để việc truy cập của người dùng sẽ dễ dàng hơn

- Ta có thể để thêm các thông tin, thông báo về BI và data warehouse để luôn có được thông tin cần thiết một cách sớm nhất

=>> Vậy làm thế nào để xây dựng một BI portal? Ở dạng đơn giản nhất thì nó là một web page chứa nhiều đường link tới các report và ứng dụng BI

=> Ta có thể dùng ASP.NET để build chúng, với tài liệu, hướng dẫn người dùng và các thông tin hữu ích khác được thêm vào thì nó sẽ trở thành một điểm khởi đầu thuận lợi cho BI users

- Về việc cài đặt BI trên SQL Server platform thì ta có thể build BI portal sử dụng SharePoint để tận dụng ưu điểm của việc tích hợp Reporting Services web parts với SharePoint

## Summary

- Có một số dạng của ứng dụng BI như là reporting, analytics, data mining, dashboards và alerts

- Ta đã thảo luận về BI portals, thứ sẽ unify toàn bộ các ứng dụng trên trong một nơi trung tâm và đơn giản hoá việc quản trị về security

- Toàn bộ các ứng dụng BI sẽ đưa đến cho ta những thông tin về điều kiện của doanh nghiệp. Ta cần cài đặt các hành động chính xác để cải thiện hiệu quả của doanh nghiệp và xác minh rằng hành động đó là chính xác bằng cách xem xét lại các ứng dụng BI

=>> Nếu hiệu quả được cải thiện thì hành động của ta là chính xác và ngược lại

# Using Data Warehouse for Customer Relationship Management

- Customer Relationship Management (CRM) là hoạt động tạo dựng liên lạc và quản lý các kết nối với khách hàng, phân tích thông tin về khách hàng, tiến hành các chiến dịch để thu hút khách hàng mới, triển khai các business transaction với khách hàng, phục vụ khách hàng và cung cấp các hỗ trợ tới khách hàng

- Một ứng dụng CRM là hệ thống mà công ty hay tổ chức sử dụng để tạo và gửi các chiến dịch, phân tích người dùng cá nhân và thông tin demographic, phân tích và dự đoán hành vi khách hàng bao gồm purchases, perform customer scoring, manage customer permissions, manage customer support/service, integrate customer data, manage customer loyalty programs, and manage customer contacts (including complaints and resolutions)

- Các ứng dụng CRM được host ở bên trong (in-house) hoặc bên ngoài với một ASP (application support provider)

- Customer data có thể được lưu trữ trong data warehouse hay trong ODS nơi mà customer data được tích hợp để tạo ra single-customer view (SCV)

- SCV là một tập đã được tích hợp của customer data từ nhiều data source khác nhau trong một địa điểm trung tâm và ta sẽ có một holistic view về khách hàng của mình => SCV rất quan trọng bởi chúng đem đến cho ta những cái nhìn và hiểu sâu hơn về khách hàng

- Một lợi ích to lớn khác của việc lưu trữ integrated customer data trong data warehouse đó là để biểu diễn việc phân tích khách hàng và data mining, bao gồm các descriptive analytics và predictive analytics

=>> Dimensional data warehouse là một platform lý tưởng cho việc biểu diễn customer scoring (tính toán các giá trị của khách hàng sử dụng công thức cụ thể), đặc biệt là khi các giải thuật cho scoring tham gia vào việc phân tích demographics và purchase

## Single customer view

- Một concept của SCV đó là đem đến một định nghĩa đơn lẻ về việc “who a customer is?”

=> Chỉ khi ta có được một định nghĩa độc lập giống nhau qua tất cả các phòng ban thì ta mới có thế có được một definitive list of customer

- SCV là một tập đã được tích hợp của customer data từ nhiều data source khác nhau trong một địa điểm trung tâm và ta sẽ có một holistic view về khách hàng của mình

- Đối với mỗi phòng ban khách nhau thì tập khách hàng cũng sẽ khác nhau, họ có thể là người mua, người dùng đăng kí, người theo dõi…

- Sẽ có hai thứ ta cần phải xem xét khi mà tạo ra một SCV:

1. Ta có thể định dạng, nối và lặp khách hàng như thế nào và ta cần phải đồng ý với các quy định của doanh nghiệp cho việc đó
2. Thuộc tính khách hàng

- Ta có thể có một csdl khách hàng trung tâm chứa personal information, their orders, their preferences, their quotes, their e-mail promotions, their downloads, their contact details, their interests, their subscriptions, their complaints, their queries, their credit score, their loyalty score, their browsing/shopping history, their account details, their competition participations, their payments, their registration details, and their security details all in one central place

- Một SCV rất quan trọng vì chúng đem đến cho ta một sự nhìn nhận và hiểu sâu về khách hàng của mình. Khi ta kết hợp data từ nhiều hệ thống khác nhau thì ta sẽ có một góc nhìn hoàn chỉnh về khách hàng. Ta càng hiểu rõ về khách hàng thì ta càng có thể tailor các sản phẩm và dịch vụ phù hợp với sở thích và mong muốn của họ, từ đó việc kinh doanh sẽ thành công => ***SCV is about an integrated view of our customers from various sources***

🡺 Ưu điểm của việc có một SCV trong data warehouse so sánh với ODS hay CDI là:

* Slowly changing dimension (SCD) => Ta sẽ có nhiều historical information => data richness
* Analytical nature of a dimensional data warehouse => cho phép ta có thể biểu diễn việc phân tích khách hàng được tốt hơn, điều này dẫn đến sự hiểu biết tốt hơn về khách hàng => Dimensional data warehouse là một nơi lý tưởng cho việc biểu diễn các phân tích nhu là việc phân tích hành vi tiêu dùng của khách hàng hay subscription dynamics để lấy được customer score và subscription band
* Computational power of a data warehouse system => Hệ thống data warehouse được thiết kế và xây dựng để có thể xử lý được các yêu cầu xử lý data khối lượng lớn và có thể tiến hành các truy vấn vào các data set rất lớn một cách hiệu quả
* Customer data trong data warehouse được tích hợp từ nhiều source system nên cho phép ta có thể kết hợp các thuộc tính của khách hàng từ nhiều nguồn khác nhau

## Campaign segmentation

- Một campaign là một kết nôi được gửi tới khách hàng một cách thường nhật thông qua e-mail, post, RSS, or text messages and containing marketing promotions (cross-sale, up sale, and so on) or notifications.

- Một campaign là một kết nối gửi tới customers

- Một delivery channel là kênh phương tiện được dùng để gửi campaign

- Open rate và click-through rate là các thông tin được sử dụng trong e-mail campaigns. Open rate là số lượng của tin nhắn e-mail được mở bởi khách hàng trên tổng số tin nhắn được gửi. Click-through rate hay CTR là số lượng khách hàng bấm vào đường link được nhúng vào trong tin nhắn e-mail trên tổng số tin nhắn được gửi

- Delivery rate là số lượng các tin nhắn e-mail được chuyển đến thành công trên tổng số tin nhắn e-mail được gửi

- Bounce rate là số lượng các tin nhắn e-mail mà không được chuyển đến cho người nhận trên tổng số tin nhắn e-mail được gửi

- Một campaign là fundamental cornerstone of CRM activity và quá trình tạo ra một chiến dịch CRM là:

* Dựa vào mong muốn của doanh nghiệp, ta quyết định khách hàng nào ta muốn hướng đến và số lượng khách hàng đó là bao nhiêu
* Tạo ra một truy vấn lựa chọn mà lựa chọn từ một csdl, một danh sách các khách hàng phù hợp với tiêu chuẩn chi tiết
* Soạn thảo nội dung chiến dịch, từng cái riêng cho mỗi kênh truyền và nền tảng
* Tổ chức chiến dịch trong hệ thống quản lý chiến dịch, dán vào truy vấn lựa chọn, tải content file, thêm nhóm điều khiển, triển khai chúng, gửi tới cho testers, lấy các chấp thuận cuối, triển khai chúng và gửi tới customer
* Giám sát, đánh giá các chỉ số, tỉ lệ và kết quả tăng trưởng doanh thu từ chiến dịch

- Một chiến dịch sẽ hướng để một danh sách khách hàng cụ thể, ta gọi đó là một segment (bộ phận, phân đoạn)

- Ta sẽ có bốn dạng của truy vấn lựa chọn:

* Permission
* Demographic data
* Customer behavior
* Campaign response

- Lợi ích của việc sử dụng data warehouse cho campaign selection và segmentation? Nó nhanh, và dễ dàng hơn, cùng với đó là về capacity

- Một truy vấn phân đoạn của chiến dịch thông thường có thể chạy mỗi lần một tuần khi nó được vận hành

## Permission management

- Ta phải gửi các chiến dịch chỉ tới những người mà ta đã xác định từ trước. Sự bằng lòng này được hiểu như là permission

- Duy trì permissions là phần quan trọng nhất trong việc vận hành CRM bởi gửi một chiến dịch đến mà không có sự cho phép của người dùng đồng nghĩa với việc ta đang spamming

- Có bốn dạng customer permission:

* General permission => Trong loại này, khách hàng sẽ bằng lòng với việc ta sẽ gửi tới họ bất kì các kết nối liên quan đến marketing
* Specific permission => Trong trường hợp này, khách hàng đồng ý cho chúng ta gửi các dạng cụ thể của communication
* Subscription => Trong dạng này, khách hàng đăng kí một chiến dịch cụ thể => Ta có thể gửi họ chỉ những chiến dịch mà họ yêu cầu cùng với các kênh truyền được đăng kí trước đó
* Negative subscription => Với dạng yêu cầu này thì khách hàng không muốn nhận bất cứ chiến dịch nào và sẽ được liệt vào danh sách ban

🡺 Lợi ích của việc có data warehouse cho việc tiến hành triển khai permission là gì? Permission được dùng trong truy vấn lựa chọn và dimensional data warehouse đem đến hiệu quả cao khi biểu diễn các truy vấn lựa chọn

- Subscription management là quá trình giao dịch và được capture bởi ứng dụng front-end

- General permission is also transactional in nature và được capture khi khách hàng đăng kí trên web ste hay khi trở thành khách hàng

- Khi ta đang ở trong chủ đề về quản lý sự cho phép => ta sẽ cần nói đến việc opting out hay revoking permission. Ta sẽ cần rõ ràng rằng là khách hàng có thể huỷ đăng kí hay bỏ sự cho phép của họ đối với chiến dịch mọi lúc => It is important that the propagation of unsubscription information to the CRM data warehouse is frequent, ideally intraday or at least once a day, so that we can make sure customers who have revoked their permissions do not get subsequent campaigns

## Delivery and response data

- Campaign delivery data là việc có hay không chiến dịch mà ta gửi được vẫn chuyển đến người nhận được xác định từ trước

- Campaign response data là việc phản hồi từ khách hàng sau khi nhận được chiến dịch của ta

- The number of e-mail messages sent out, the number of messages delivered, the number of messages not delivered (bounced), and the number of e-mail messages not sent out (either because of banned list or bounce list) được gọi là campaign delivery data

- The number of e-mail messages opened/read, the number of messages clicked by the customers, and the number of customers purchased some products được gọi là campaign response data

=> Cả 2 đều quan trọng bởi chúng biểu thị sự thành công hay thất bại của chiến dịch

- Tỉ lệ open và click-through khác nhau giữa các ngành với nhau => sẽ rất hữu ích khi có thể hiểu được làm thế nào để campaign delivery and response data được thu thập bởi hệ thống CRM

- The benefit of using a data warehouse (as opposed to an operational system such as an ERP, CDI, or ODS) is that we can use the delivery and response data for campaign segmentation. This is because the delivery and response data will be located on the same platform as the other data used for a selection query (for example, customer data, shopping/purchase/activity data, and permission data) during the campaign selection and segmentation process

- Lợi ích thứ hai đó là ta có hiệu suất tốt hơn khi tải data và biểu diễn phân tích => cực kì đúng khi truy vấn campaign response data cho việc phân tích khách hàng và phân tích mua hàng

- Campaign delivery data cho phép ta phân tích bounce rate qua các miền e-mail khác nhau và hiểu lý do cho tỉ lệ đó

## Customer analysis

- Ta có thể sử dụng CRM data trong data warehouse cho việc phân tích hoạt động và hành vi khách hàng => customer analysis

- Có hai dạng chính của việc phân tích khách hàng đó là descriptive analysis (cố gắng hiểu hay mô tả hành vi và hoạt động của khách hàng) và predictive analysis (dự đoán hành vi và hoạt động của khách hàng)

- Một CRM data warehouse chứa customer data, permission data, campaign response data, customer activity data (purchase, shopping, subscription, gain, and loss), billing data, payments data, and marketing data (including promotions and campaigns)

- Một CRM data warehouse rất hữu ích cho việc phân tích người dùng, phương pháp để làm việc phân tích này sẽ giống với those of BI như là reporting, analytics và data mining

## Customer support

- Hầu hết các hoạt động hỗ trợ khách hàng được hoàn thành trong các hệ thống vận hành, gồm cả call center systems và ERPs

- Các customer service agent thường xuyên sử dụng hai dạng này của hệ thống để hỗ trợ và phục vụ khách hàng

- ODS cũng được sử dụng ngày càng nhiều cho việc hỗ trợ khách hàng

- Các phương pháp cơ bản được sử dụng để vận chuyển những chức năng hỗ trợ khách hàng đó là analytics và data mining. Một số chức năng còn yêu cầu về notification và subscription

## Personalization

- Cá nhân hoá là việc customization của web sites, campaigns, offers, products và services tới khách hàng cụ thể

- Customization thường được dùng cho web site thương mại điện tử để tạo các sản phẩm và đề xuất chi tiết hơn tới các khách hàng cụ thể

- Cá nhân hoá rất quan trọng bởi chúng làm gia tăng khả năng cho việc mua sản phẩm của khác hàng => gia tăng doanh thu

- Source data có thể được phân tích bằng việc sử dụng mô hình khai phá dữ liệu để score một danh sách các đề xuất và sản phẩm đối với từng cá nhân khách hàng => Output được gói lại như XML và được cho đến home pages được cá nhân hoá trên web site

- Key to site content personalization and recommendation đó là định dạng được khách hàng => Hai hướng tiếp cận được sử dụng nhiều đó là dùng logins và cookies:

* Log in (self-authentication) đáng tin và chính xác, tuy nhiên gây ra sự bất tiện cho khách hàng
* Cookies thì thuận tiện nhưng lại không đáng tin bởi nó không định dạng được được khách hàng mà nó chỉ như là một cơ chế lưu trữ thông tin của khách hàng => We still need to get a form of identification from the customer and then store this information as a cookie. Whether we use cookies or logins or both, we need to be able to identify the customer to be able to tailor the site content for personalization

## Customer loyalty scheme

- Trong rất nhiều doanh nghiệp, đặc biệt là các doanh nghiệp thương mại thì customer loyalty scheme là một hoạt động CRM quan trọng

- Customer loyalty scheme là một chương trình tôn vnh khách hàng phụ thuộc vào đóng góp của họ cho doanh nghiệp. Đóng góp ở đây được tính toán theo loyalty scores

🡺 Mục đích chính của loyalty scheme là khuyến khích khách hàng ở lại với chúng ta và sau cùng đó là làm gia tăng doanh thu và lợi nhuận

- Calculated loyalty scores có thể được lưu trữ như là một thuộc thuộc trong trường customer => makes it easier for the scores to be used in campaign segmentation for sending messages to customers as well as for analysis, và scores sẽ được quy đổi sang bands (hay classes)

- Customer loyalty scheme yêu cầu một số báo cáo và phân tích dự đoán như churn analysis và revenue analysis để có thể tính toán mức độ thành công của chương trình

=> Churn analysis phân tích số lượng khách hàng mới mà ta có được và số lượng khách hàng ta mất đi sau mỗi khoảng thời gian cụ thể

=> Revenue analysis biểu thị các doanh thu thêm được sinh ra bởi từng band hay class, so sánh với nhóm điều khiển

=> Nhóm điều khiển (control group) chứa các khách hàng mà không nhận được bất kì sự tôn vinh nào từ loyalty scheme

- Các khách hàng mới được thêm vào lược đồ một cách định kì và loyalty scores cần được cập nhật thường xuyên cho toàn bộ khách hàng => Ta cần giữ permission cho loyalty scheme luôn được cập nhật, bao gồm cả opt-ins, opt-outs và unsubscriptions

## Summary

- Ta đã trao đổi về việc thi hành data warehouse cho việc quản lý mối quan hệ khách hàng, bao gồm các chủ đề về permission, segmentation, response data, customer analysis, customer support, personalization và loyalty scheme

- Ta luôn cần giữ cho permission được cập nhật (điều quan trọng nhất trong CRM) bởi nếu không ta sẽ bị coi là đang spamming

- Campaign delivery and response data là điều quan trọng nhất tiết theo bởi nếu tỉ lệ vận chuyển mà thấp thì ta sẽ không tiếp cận được với nhóm người nhận đã được xác định trước đó và data phản hồi ước tính được sự thành công của chiến dịch

- Phân tích khách hàng cho phép ta có thể hiểu hoạt động hiện tại của khách hàng dự đoán hành vi của họ => Có hai dạng phân tích chính đó là descriptive analysis và predictive analysis

- Cá nhân hoá cho phép ta có thể tailor sản phẩm và dịch vụ, web sites và các đề xuất thông qua profile của khách hàng => Khả năng định dạng khách hàng chính là key to on-site personalization

- Trong customer loyalty scheme thì ta tính toán loyalty scores để hiểu được giá trị khách hàng và phân tích sự cải thiển trong customer churn và revenue

# Other Data Warehouse Usage

- Ta đã thảo luận về việc cài đặt data warehousing cho BI và CRM

- Ở chương này ta sẽ thảo luận về data warehousing ở trong ngữ cảnh của customer data integration (CDI), unstructrured data và role of search trong data warehousing

- Ta sẽ không nói “implementing data warehousing for CDI bởi vì data warehouse là consumer của CDI

## Customer data integration

- Trong rất nhiều tổ chức thì customer data nằm rải rác qua nhiều hệ thống hay ứng dụng khác nhau

- CDI là một ongoing process của việc retreiving, cleaning, storing, updating và distributing customer data:

• Retrieving means extracting the customer data from the source system.

• Cleaning means correcting incorrect data and removing duplicate records.

• Storing means putting the customer data in a central customer data store.

• Updating means modifying the customer data in the customer data store, according to the new values from the source system.

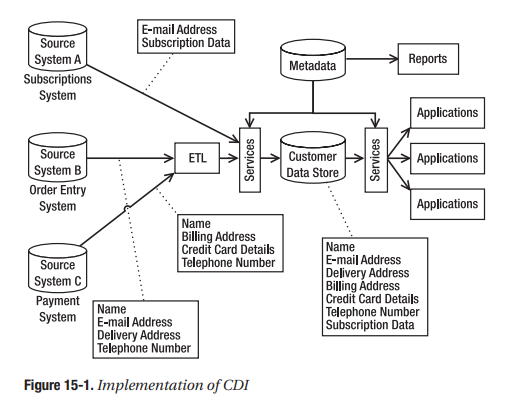
• Distributing means publishing the customer data so that other systems can use this customer data.

- CDI cho phép ta có được một phiên bản đáng tin cậy, có thể hoạt động độc lập và được dọn dẹp tốt hơn của customer data mà các ứng dụng trong doanh nghiệp có thể dùng, bao gồm các lợi ích sau:

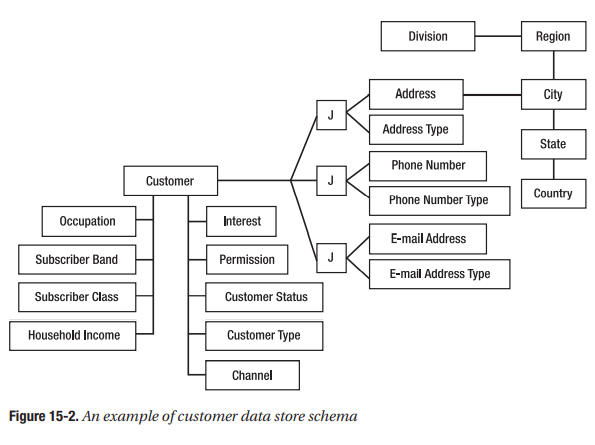
* Tăng sự hài lòng của khách hàng
* Các phân tích nghiệp vụ, doanh nghiệp tốt hơn
* Làm giảm độ phức tạp của các tiến trình, quá trình sử dụng customer data

- CDI được cài đặt sử dụng kiến trúc hướng dịch vụ, ví dụ như hình dưới đây:

- Ta có thể thấy một số source system đẩy customer data lên customer data store bằng việc gọi trực tiếp services, nhưng có sẽ có source system lưu trữ customer data chỉ ở trong database của chúng => ta có thể build ETL lấy và cleanse customer data từ source system và gọi services để lưu trữ data lên customer data store



- Customer data store có dạng normalized relational model => cho hiệu suất tốt hơn trên toàn bộ dimensional model khi được dùng để lấy, để tạo hay update một bản ghi độc lập hay là một phần của bản ghi



- Trong một số trường hợp ở trong customer data store, bạn sẽ cần lưu trữ các giá trị lịch sử của các thuộc tính customer, bằng các sử dụng một chuỗi các bảng audit. Trước khi thay đổi các thuộc tính này thì ta sẽ lưu trữ các bản ghi ban đầu với các giá trị cũ của thuộc tính trong một bảng audit cùng với timestamp lúc mà thay đổi diễn ra

## Unstructured data

- Thông thường, ta sẽ phân loại data thành hai nhóm lớn:

* Nhóm đầu tiên là numbers, character và dates => được lưu ở trong data hay database, trong các bảng mà có thể truy vấn và sắp xếp => structured data
* Nhóm thứ hai thì không được xem xét kĩ như structured data, tức là nó là dạng data đa phương tiện như images, videos, web pages, presentations, telephone conversations, e-mails, documents, và music => unstructured data => được lưu trong file servers, database servers, FTP servers, e-mail servers, content management systems (CMSs), and document management systems (DMSs).

- Khối lượng unstructured data nhiều hơn structured data rất nhiều

- Trong vào 15 năm trở lại đây của data warehouse và hầu như là cả lịch sử của database thì chúng ta đang tập trung vào structured data. Nhưng từ khi khối lượng của unstructured data nên cũng dễ hiểu khi chúng ta sẽ tìm hiểu nhiều hơn về unstructured data

- Vậy ta lưu trữ unstructured data trong data warehouse như thế nào?

=> Theo cách thường thì ta sẽ lưu chúng dưới dạng files trong hệ thống file và lưu trữ các thuộc tính và links trong data warehouse

- Ta có thể lưu trữ toàn bộ các thuộc tính ở trong data warehouse và ở trong relational tables. Sau đó là lưu trữ đường đi tới file đó hay URL trong một cột trong một bảng giống như vậy để ứng dụng có thể lấy được file và truy cập vào

- Ta có thể lưu trữ các thuộc tính và links tới files trong dimensional format cho mục đích phân tích

- Ta có thể lưu trữ các document như là các đối tượng nhị phân hay đối tượng bên trong database

- Trên là phương pháp cũ đã có trên thị trường từ lâu và được xem là phương pháp truyền thống => Chúng ta có một phương pháp mới hơn, thay vì lưu trữ documents ở trong data warehouse thì ta sẽ trích xuất ra structured data từ unstructured data và sau đó lưu trữ data đã được trích xuất trong data warehouse

- Với documents thì phương pháp này có tên là text analytics => phương pháp mà ta áp dụng cho documents thì hoàn toàn có thể áp dụng cho các dạng khác unstructured data như là images hay audio

- Khía cạnh metadata của unstructured data conversion cũng rất ý nghĩa để nhắc đến. Toàn bộ các thuộc tính đã đề cập có thể được lưu trữ như là output của text analytics hay mining và được lưu trữ như là metadata trong data warehouse

- Metadata được lấy bởi các ứng dụng client thông công cụ tìm kiếm => Sẽ rất quan trọng để giữ cho metadata luôn được chính xác và được cập nhật liên tục

- Một thứ khác của unstructured data trở thành một ứng dụng là email store

## Search in data warehousing

- Khi mà data warehouse phát triển đến khối lượng TB và có hàng trăm báo cáo, cubes và rất nhiều BI data models thì việc có một công cụ tìm kiếm sẽ giúp hỗ trợ users trong việc tìm ra vị trí chính xác của các báo cáo, cubes và data models

- Một search facility là một công cụ đánh chỉ mục data warehouse reports, analytics hay data models (cả metadata và content) cho phép users tìm ra thông tin bằng việc gõ các search criteria trong search text box

- Một search facility là có thể sắp xếp độ ưu tiên của các kết quả được hiển thị đến theo mức độ liên quan

- Một typical search application bao gồm một indexer, retriever và scoring routines:

* Indexer browses the folders for all of the file types (unstructured data files, and so on), then opens and crawls each file, and then stores the locations of those words in the repository. The indexer also collects the information about the reports, cubes, and data models and stores this information in the repository. It also stores previous searches and queries.
* Retriever accepts a value from the search text box, searches its repository and metadata, and displays the results to the end users. Before it is displayed to the end users
* Scoring routine sorts the results according to a preset logic/algorithm. The scoring routine may also partition the result into more than one section to be displayed separately on the screen

- Search sẽ đi sâu vào report metadata, multidimensional metadata, data warehouse metadata và BI data model metadata để tìm ra các report liên quan, cubes và models => Có thể tìm kiếm content của stored reports và lịch sử tìm kiếm trước

- Công cụ tìm kiếm có thể tìm ra các báo cáo liên quan và làm nổi bật chúng cho BI users

- Các tìm kiếm trong quá khứ là những gì users đã tìm kiếm trong search box mà giúp đẩy nhanh quá trình tìm kiếm

- Thay vì cố gắng xây dựng một công cụ tìm kiếm riêng cho data warehouse thì việc install và implement một công cụ tìm kiếm từ BI vendors => BI hay data warehouse search là specialized area

- Có 3 areas trong data warehousing sẽ có lợi ích từ công cụ tìm kiếm:

* Đơn giản hoá giao diện người dùng => Các kết quả tìm kiếm có thể phân hạch giữa các kết được tìm kiếm trong metadata area và kết quả được tìm kiếm trong actual data
* Reports search
* Unstructured data => documents, webpages, text files, images, videos and audio files

## Summary

- Chương này ta đã tìm hiểu về việc cài đặt của data warehouse với ngữ cảnh của CDI, unstructured data và search

- Bên cạnh đó, ta có có một vài cài đặt khác của data warehouse cho những ngành khác nhau như research/science/labs, geology, topology, earth science, astronomy, particle physics…

# Testing Your Data Warehouse

- Như các hệ thống IT khác thì sau khi xây dựng data warehouse với ETL và applications của chúng thì ta cần phải kiểm thử thông suốt các quá trình. Ta cần phải tiến hành 6 dạng kiểm thử đó là:

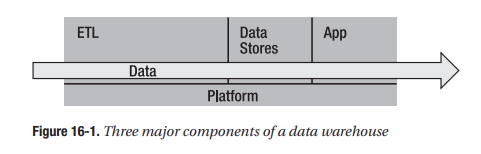
* ETL testing
* Functional testing
* Performance testing
* Security testing
* User acceptance testing
* End-to-end testing

- Ta sẽ tập trung vào cơ chế và quy trình của từng loại kiểm thứ, chúng sẽ được tiến hành như thế nào trong ngữ cảnh của data warehouse

## Data warehouse ETL testing

- ETL testing rất quan trọng bởi vời ETL đem data từ source system lên data warehouse => nếu ETL diễn ra không chính xác thì data trong data warehouse sẽ bị sai và dù cho data model hay applications có tốt như nào thì cũng không thể sử dụng được

- Data warehouse system có ba thành phần cơ bản đó là ETL, data stores và applications => Trong 3 thành phần này thì ETL luôn đóng góp nhiều nhất vào cả quy trình vận hành của data warehouse (~ 50%), tỉ lệ này phụ thuộc vào độ phức tạp của applications



- Nhiệm vụ chính của ETL testing như sau:

* Đảm bảo bạn có thể lấy được toàn bộ data mà ta cần, không bị mất data updates trong source systems
* Đảm bảo data được tải lên đúng đến data warehouse, đến đúng bảng, đúng cột, đúng format và đúng thời gian
* Đảm bảo ETL gia tăng làm việc đúng theo thiết kế dù là sử dụng kiến trúc nào
* Đảm bảo các bulk load script load chính xác data lên data warehouse

- Ta tiến hành xây dựng việc kiểm thử ETL. Bắt đầu từ sở đồ biểu đồ kiến trúc ETL và các quá trình ETL. Ta sẽ cần test cả quá trình ETL tổng thể và các task riêng lẻ vân chuyện data đến chính xác ở data store

- Một đặc tính quan trọng để kiểm thử đó là recoverability => có nghĩa là các quá trình ETL cần đủ mạnh để trong một sự cố hay lỗi nào đó hệ thống có thể tự chạy lại mà không bị thất thoát và làm hỏng dữ liệu. Các quá trình ETL cần được xây dựng để nếu như quy trình bị dừng ở bất cứ điểm nào và khi tiến hành chạy lành thì sẽ không có data bị mất

- Với kiến trúc hướng dịch vụ (SOA architecture) sử dụng trickle feed thì ta cần giả lập lỗi và lỗi mạng lưới hàng đợi tin nhắn (MQ failure or network failure), nơi mà tin nhắn được gửi từ source system nhưng mà không đến được hệ thống đích và cần được gửi lại

- Ở trong một số project thì ta có thể sử dụng kiến trúc ETL gia tăng để tiến hành tải các thứ cồng kềnh trước (initial bulk loading) lên data warehouse

- Trong các project khác thì ta có thể sử dụng kiến trúc ETL phân chia (separate ETL architecture) cho việc tải lượng lớn (bulk loading) bởi ETL gia tăng không được thiết kế để xử lý việc load dữ liệu qúa lớn => Sẽ rất là thiết yếu để kiểm thử xem ta có bị mất data ở cutoff data không khi có một potential gap giữa bulk load cutoff date và start date of the incremental load

## Functional testing

- Kiểm thử chức năng (functional testing) là việc đảm bảo rằng toàn bộ yêu cầu của doanh nghiệp được thoả mãn. Business requirements là data và sự khả thi mà users cần từ data warehouse để có thể tiến hành các business tasks

- Business requirements được xác định ngay từ thời điểm đầu của một dự án data warehouse

- Trong việc kiểm thử chức năng thì ta cần kiểm tra toàn bộ business requirement mà ta đã xác định ở thời điểm của dự án, một trong số kĩ thuật để kiểm thử đó là verify the total

- Sẽ rất quan trọng khi mà tester hiểu được cách để verify data warehouse figures. Họ sẽ so sánh data warehouse figures với source system, hoặc là truy vấn csdl source system hay chạy một số các báo cáo trên các ứng dụng source system. Tester cần biết cách để sử dụng các source system để lấy ra các data warehouse figure được test

- Functionality testing khôgn chỉ là tính toán các tổng hợp (totals) và so sánh các phép đếm (counts) giữa source system và data warehouse. Ta cần kiểm thử SCD (type 1,2 hay 3), kiểm thử surrogate keys để xem nó có kết nối bảng fact với các bảng đim thích hợp không, kiểm thử sự toàn vẹn tham chiếu trong ODS, kiểm thử dimension, kiểm tra dimension hierarchies, kiểm thử toàn bộ thuộc tính, xác minh toàn bộ các chiều degenerate trong tất cả các bảng fact, kiểm thử toàn bộ các cột điều khiển, kiểm thử tất các bộ lọc về data quality đang hoạt động chính xác và cuối cùng đó là kiểm tra metadata gồm 7 loại và cần chắc chắn rằng là chúng được điền hợp lý

- Các viễn cảnh doanh nghiệp và điều kiện của doanh nghiệp cũng cần được kiểm thử trong functionality testing

- Test data rất quan trọng trong việc tiến hành kiểm thử chức năng. Ta sẽ cần rất nhiều nỗ lực để tạo ra test data cho một test condition cụ thể. Không giống như kiểm thử ứng dụng khác thì trong data warehouse functional testing ta cần tạo các test data ở trong source system chứ không phải data warehouse

## Performance testing

- Kiểm thử hiệu năng (performance testing) xác minh rằng là tất cả các yêu cầu về kĩ thuật liên quan đến platform, capacity, latency và response time được thoả mãn

- Performance testing rất quan trọng bởi khi phát triển các quá trình ETL thì ta chỉ dùng một khối lượng nhỏ data và thường chỉ tập trung vào việc đảm bảo các chức năng của quá trình ETL là chính xác => Khi các chức năng đã được phát triển xong thì đó là lúc cần vận hành chúng trong điều kiện bình thường của doanh nghiệp, bao gồm work load, timing, amount of data loaded, and amount of data already existing in the target tables, hardware, indexes, partitions, number of users, type and amount of queries, cube-processing times, number of variables in data mining and predictive analysis…

- Có hai area chính để kiểm thử hiệu năng trong một data warehouse:

* First area đó là ETL. Trong ETL area ta cần xác minh cả hiệu năng tải hàng loạt (bulk load performance) và hiệu năng tải gia tăng có thoả mãn tiêu chuẩn về hiệu năng đã đề ra hay không. Bulk load được dùng để điền vào các data warehouse lần dầu tiên, còn incremental load được dùng để điền vào data warehouse khi nó đã được vận hành
* Second area là các ứng dụng => tiêu chuẩn về hiệu năng bao gồm các thời gian phản hồi

🡺 Mục đích của việc tiến hành kiểm thử hiệu năng đó là làm hài lòng yêu cầu về hiệu năng

=>> Một số yêu cầu thì áp dụng cho toàn bộ hệ thống data warehouse nhưng một số chỉ áp dụng cho một số bộ phận cụ thể trong hệ thống

- Vậy có những task gì ta cần làm để tiến hành kiểm thử hiệu năng của data warehouse?

=> Có ba bước chủ yếu để tiến hành trên các quá trình ETL:

* Configure the test environment
* Run the bulk load processes
* Run the incremental processes

- Các bước ở trên là về việc kiểm thử cho ETL. Đối với applications thì để ước tính hiệu năng ta có thể ghi lại, gõ lại các actions hay ước tính nó một các thủ công => Scripting the actions có nghĩa là ta sẽ dùng phần mềm để ghi lại các hoạt động mà một user làm khi sử dụng applications

- Trong application performance testing, ta sử dụng applications, nó có thể là reports, mining models cubes hay BI applications, sử dụng cả normal scenarios và exception scenarios

- Như đã thảo luận ở chương 6 thì có một số cách ta có thể tiến hành để cải thiện hiệu năng của application là sử dụng summary tables, indexing và partitioning tables

- Ta cần suy nghĩ về hiệu năng khi ta thiết kế data warehouse chứ không phải khi đã hoàn thành việc build chúng. Ta cần bắt đầu tính đến hiệu năng khi mà ta thiết kế architecture và khi xây dựng architecture => Toàn bộ mọi thứ nên được thiết kế cho hiệu năng ở mọi bước đơn lẻ từ khi bắt đầu dự án chứ không phải là hệ thống data warehouse được xây dựng xong

## Security testing

- Kệ thử bảo vệ (security testing) là việc đảm bảo chỉ có những người và ứng dụng được phép truy cập vào data warehouse là có thể truy cập được chúng trong thực tế. Bên cạnh đó, ta cần kiểm thử cách mà mọi người truy cập vào data trong data warehouse như là dùng tool hay cách user cung cấp thông tin xác thực bảo mật

- Một data warehouse bao gồm một số data store, ở trong mỗi data store ta có một số data objects như tables, views và stored procedures => Yêu cầu về bảo vệ nêu lên rằng users nào nên có thể truy cập vào objects nào. Ta xác minh các yêu cầu bằng cách kết nối tới data warehouse như là users được phép truy cập và truy cập vào các objects bằng kết nối của họ, đồng thời cố gắng truy cập vào các object giống như vậy (You verify these requirements by connecting to the data warehouse as those users and accessing those objects and by connecting as other users and trying to access the same object)

- Ta kiểm thử data warehouse security bằng việc truy cập data warehouse từ các ứng dụng như reports hay BI tools và bằng việc tiến hành các truy vấn SQL trực tiếp tới data stores

- Ta cũng xác minh xem là security audit table được điền vào chính xác và hợp lý, chứa username và thời gian họ truy cập vào data warehouse, bằng cách so sánh các hoạt động ta làm trên ứng dụng và security logs

## User acceptance testing

- Ba quá trình kiểm thử ở trên được tiến hành bởi team dự án phát triển data warehouse. Một khi mà ba quá trình trên hoàn thành thì sẽ đến lượt end users kiểm thử data warehouse => đó là user acceptance test (UAT) => Đây là nơi một số key user sử dụng data warehouse và các ứng dụng của chúng để tìm ra xem có hay không chúng đã fullfill những gì mà ta cần, đồng cũng kiểm thử sự thân thiện với người dùng (user friendliness)

- Mục đích của việc tiến hành UAT có hai hướng:

* Mục đích cơ bản đó là xác nhận xem hệ thống có khả năng thực hiện các tasks như đã được thiết kế không.
* Mục đích thứ hai đó là có thể lấy được feedback từ user, đặc biệt là về user friendliness

- Trong UAT, ta xác nhận khả năng chứa của hệ thống để thực hiện các daily business task và đáp ứng nhu cầu của users. Test cases trong functionality testing được thiết kế để cover toàn bộ các khả năng, dựa trên sự cụ thể về chức năng và phi chức năng, bắt nguồn từ business requirements

- Trong kiểm thử chấp nhận người dùng, một cách lý tưởng thì các test case được thiết kế để phản ánh hoạt động hàng ngày của nhiều role khác nhau ở trong user base => it’s about how easy it is for the business users from each role to use the data warehouse and its applications to perform their tasks

- User friendliness của các ứng dụng giống nhau sẽ có sự khác nhau giữa user này và user kia bởi mỗi người có trách nhiệm khác nhau

## End-to-end testing

- Đến đây, data warehouse đã sẵn sàng để go to production. The ETL tasks have been verified in the ETL test. All the functionalities has been tested in the functionality test. The performance issues have been sorted out in the performance test. The security has been checked in the security test. The users have reviewed the data warehouse, and all their requests have been processed

=> Trước khi triển khai data warehouse into production thì sẽ còn một quá trình kiểm thử nữa cần phải làm đó là end-to-end testing, nơi mà ta xác minh toàn bộ hệ thống từ đầu đến cuối

- Để tiến hành kiểm thử end-to-end thì ta sẽ để data warehouse ETL processes chạy một vài ngày để giả lập các điều kiện vận hành, các ETL batch cũng sẽ chạy tự động. Nó giống như là tiến hành final check của mọi thứ nhưng lần này ETL chạy tự động, feed data tới data warehouse mọi lúc và theo lịch trình

- Một số người gọi quá trình này là kiểm thử tích hợp

- Khi ta tiến hành quá trình kiểm thử này thì ta cần phải viết tài liệu về mọi thứ: all the details regarding the configuration; all the details about the initial data; information about which code base you deployed, how to deploy them, and which cutoff date you loaded the data from; details of the security/ functional account that you use to set up... => Ta sẽ ghi chút toàn bộ và mọi thứ gì có thể bởi ta cần lặp lại việc triển khai trong production. Note này có thể coi là form của quy trình triển khai

- Sau khi đã hoàn thành quá trình kiểm thử end-to-end một cách chỉn chu và thoả mãn thì bây ta đã sẵn sàng đến với production. Với note mà đã được ghi, ta sẽ tiếp cận hệ thống production với sự cẩn trọng lớn

## Migrating to production

- Bây giờ ta đã sẵn sàng để tiến hành triển khai data warehouse đến với production. Ta sẽ làm các công việc y hệt như những gì đã làm với quá trình kiểm thử tích hợp, chỉ là giờ đây ta sẽ làm chúng trong môi trường production

- Ta sẽ execute the same scripts, copy the same executables/files, set the same configurations, load the same initial data and activate the same ETL processes => Ta sẽ làm từng bước độc lập trong quy trình triển khai ta đã tạo ra ở quy trình kiểm thử end-to-end

- Trước đó, hãy đi đến “go live” checklist, cái này thì mọi hệ thống đều có thể sử dụng được. Checklist cover các khía cạnh về kĩ thuật của việc triển khai sản xuất như environment, help desk, handover, user training và documentation

* Prepare the production environment => đảm bảo rẳng môi trường sản xuất đã sẵn sàng => Sẽ rất cần thiết để việc triển khai của data stores và quá trình ETL đến production có thể diễn ra tốt đẹp
* Coordinate with the help desk for first-line support => đảm bảo rằng các nhân viên hỗ trợ có thể hỗ trợ data warehouse và ứng dụng của chúng. Sẽ rất cần thiết để đảm bảo rằng là data warehouse users sẽ có được sự hỗ trợ cần thiết từ help desk
* Hand over the data warehouse system to the operation team => Ta transfer the knowledge và responsibility từ team phát triển (team dự án) đến team vận hành. Sẽ rất cần thiết để đảm bảo rằng khi mà team dự án không tham gia nữa (sau khi dự án kết thúc) thì users vẫn có được những hỗ trợ cần thiết cho việc giải quyết các vấn đề vận hành và yêu cầu nâng cao
* Train users and create documentation => Huấn luyện end users và tạo các hướng dẫn người dùng. Sẽ rất cần thiết để đảm bảo rằng users sẽ có kiến thức cần thiết để sử dụng các ứng dụng và truy vấn được vào data warehouse
* Liaise with the DBA => Ta trao đổi về yêu cầu quản trị csdl với DBA. Sẽ rất cần thiết để đảm báo rằng data warehouse database sẽ được hỗ trợ một cách hợp lý bởi DBA
* Create the data warehouse portal => Sẽ rất cần thiết để đảm bảo rằng users có những access points riêng lẻ đến data warehouse reports và applications cũng như là user guide và help desk numbers. Cấu hình và tổ chức data warehouse portal chính xác dựa theo cấu trúc mà users cần. Đừng quên xây dựng các sắp xếp về bảo vệ theo cùng

- Bây giờ ta đã chuânr bị mọi thứ và sẵn sàng triển khai data warehouse đến với môi trường production. Khi ta đến với production thì sẽ không quá trình kiểm thử nào nữa

## Summary

- Ta đã trao đổi về quá trình kiểm thử trong data warehouse

- Ta đã trao đổi một chút về “go live” checklist và cần phải theo dõi khi triển khai data warehouse đến production

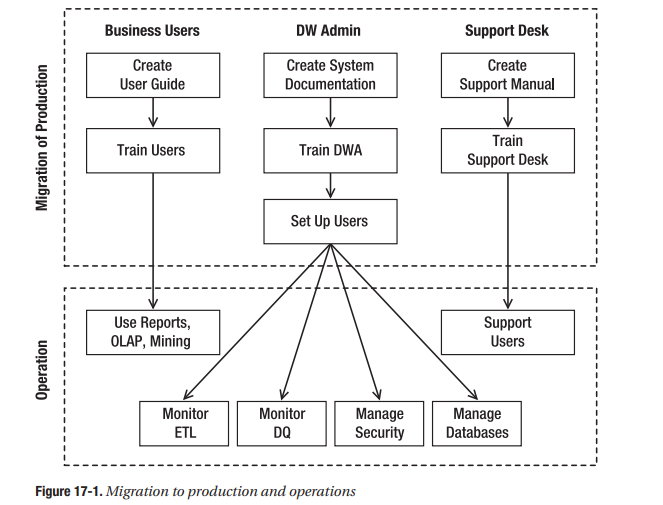
- Ta không nên đánh giá thấp quá trình kiểm thử bởi chúng sẽ mất một khoảng thời gian tương đối và nó là khoảng thời ý nghĩa để xác minh data warehouse sẽ hoạt động một cách hiệu quả

- Ta phải luôn ghi nhớ rằng kiểm thử bao gồm fixing và cần thêm thời gian để fix những lỗi hay cung cấp các yêu cầu về thay đổi

# Data Warehouse Administration

- Các nhiệm vụ của việc đưa data warehouse đến production được phần làm 3 dạng tuỳ theo là nhân viên nào tham gia:

* Đầu tiên, BI manager tạo user guide và team phát triển viết tài liệu hệ thống và hướng dẫn hỗ trợ
* Sau đó, business intelligence manager (BIM) hướng dấn business users và team phát triển hướng dẫn data warehouse administrator (DWA) và hỗ trợ nhân viên tư vấn
* Cuối cùng, DWA tổ chức cho users và data warehouse đi vào vận hành



## Monitoring data warehouse ETL

- Một trong những thứ quan trọng nhất trong quản trị data warehouse đó là giám sát các quá trình ETL. Các quá trình ETL feed data warehouse với data

- Nếu ETL hoàn toàn không chạy vào ngày hôm đó, thì điều đó cũng không tệ lắm, vì điều đó có nghĩa là dữ liệu trong kho đã cũ, vậy thôi. Nhưng nếu ETL đang chạy nửa chừng vào ngày đó thì điều đó thật tệ, vì dữ liệu có thể ở trạng thái chưa hoàn thiện. Điều đó không nhất thiết là sai, nhưng nó chưa hoàn chỉnh

=> Đó là lý do vì sao việc giám sát ETL lại quan trọng, khi kiểm tra có hay không các quá trình ETL triển khai thành công. Có một điều tối quan trọng đó là bạn phải tự động hoá quá trình giám sát ETL này và có 4 lý do cho điều đó:

* Nếu nó là một quá trình thủ công thì ta sẽ không biết được là ETL có đang chạy đúng hay không
* Ta cần tự động hoá việc giám sát ETL để khiến nó trở nên dễ dàng hơn, nhanh hơn cho quá trình giám sát
* Ta cần tự động hoá quá trình giám sát ETL bởi nó có thể sẽ fail vào thời điểm nửa đêm và không ai thông báo cho đến giờ hành chính sáng hôm sau => Điều này ĐẶC BIỆT QUAN TRỌNG nếu data warehouse được sử dụng cho CRM thay vì là BI bởi có thể sẽ gây ảnh hưởng trực tiếp đến doanh thu và doanh số của doanh nghiệp
* Ta cần tự động hoá quá trình ETL bởi vì sự nhất quán

- Một cách lý tưởng, các ETL batch không nên bị fail bởi bad data nhưng trong thực tế vẫn xảy ra điều đó. Bad data nên đi tới data quality quarantine area và không được tải lên data warehouse. Chúng sẽ tự động được correct dựa theo business rules và được tải lên data warehouse => Có 2 thứ để làm để giám sát các quá trình ETL:

* Notification => Hệ thống cần phải cho ta biết là ETL có đang hoạt động tốt hay thất bại. Đây là một quá trình giám sát data warehouse ETL tasks và thông tin đến nhân viên vận hành về trạng thái của các task này => Có 2 hướng tiếp cận hay được dùng cho notification, một là ETL tasks sẽ tự ghi lại trạng thái của chúng (rẻ hơn, có được nhiều thông tin nhưng có mức độ rủi ro hơn), hai là một hệ thống giám sát bên ngoài sẽ kiểm tra các ETL task một cách định kì (đắt hơn, có ít thông tin có thể trích xuất nhưng đáng tin cậy hơn)
* Reconciliation là khi ta so sánh data nguồn với data đích để tìm ra data ở trong data warehouse có chính xác hay không. Việc so sánh này sẽ kiểm tra các giá trị tổng hợp và các data cụ thể ở trong data warehouse có giống với data ở trong source system hay không. Quá trình reconciliation là sự so sánh mang tính hệ thống giữa data ở trong data warehouse và source system. Sự so sánh về hệ thống này có nghĩa là ta sẽ kiểm tra một dãy các giá trị tổng hợp được định nghĩa từ trước và các data cụ thể một cách định kì

- Ta sẽ cần phải cẩn thận khi mà ETL kết hợp data từ nhiều source system khác nhau sử dụng một formula cụ thể

- Quá trình reconciliation cần được tiến hành định kì như là một quá trình đang chạy để đảm bảo rằng chất lượng của data mà ta tải lên data warehouse là tốt ở mọi thời điểm

## Monitoring data quality

- Users review data quality reports và tiến hành các hành động để correct data ở trong source system

- Mặc dù business users có trách nhiệm cho việc kiểm tra sự sai phạm của data quality rules ở trong business areas của họ thì BIM cũng cần giám sát các xu hướng chung sau:

* The number of data warehouse rules violation over time
* The geographic dispersion of rule violations
* The other overall data quality analysis

- Các xu hướng của data quality cần được giám sát liên tục. Rất có khả năng rằng số lượng của các sai phạm cho một rule cụ thể nào đó có thể giảm, sau đó só sự đều đặn và cuối tăng đột ngột trở lại. Đôi khi thì sự gia tăng có thể diễn ra ở nhiều rules cùng lúc. Nguyên nhân chủ yếu của việc này đó là các hoạt động với một tác động lớn ảnh hưởng đến toàn bộ công ty hay tổ chức và từ đó ảnh hưởng đến nhiều area của data warehouse => Các sự kiện này thường được biết trước nên ta có thể lên kế hoạch và chuẩn bị trước như là sự tăng giá hàng năm, việc sáp nhập với một công ty khác hay có một source system mới feed data đến data warehouse

## Managing security

- Data warehouse security là quá trình cho phép người cần truy cập vào data warehouse và hạn chế những người không được phép truy cập, bao gồm authentication và authorization

- Authentication xác định data warehouse users là ai khi họ nói về họ, bao gồm việc xác định người dùng hỏi về việc truy cập data warehouse có được phép truy cập không => Kiểm tra cấp trên của họ hay BIM. Sau khi xác minh thì ta sẽ tạo ra role/ID/account cho họ. Ta cũng cần xoá leaver (người không còn được cấp phép truy cập nữa) và mover (người đã rời khỏi công ty) ra khỏi danh sách của data warehouse users

- Authoriztion xác định user có được truy cập tới một đối tượng data warehouse trước khi cho phép user có thể truy cập chúng => bao gồm chặn truy cập tới data stores, database, reports, cubes, mining models, portals, dashboards… ta sẽ sử dụng security groups và security roles

- Quản lý data warehouse security rất quan trọng bởi một data warehouse chứa nhiều data từ các source system khác nhau

- Vậy làm thế nào để quản lý data warehouse security? Một cách lý tưởng thì ta sẽ cần có một quá trình in place mà có thể cover việc capture các yêu cầu bảo vệ, xác minh người dùng, cấp quyền truy cập và xoá bỏ người dùng không hoạt động

- Điều quan trọng là phải nêu rõ người dùng cần quyền truy cập trong bao lâu

- DWA có thể cấp quyền truy cập cho người dùng mới. Nếu có một nhóm security cho các user giống nhau đã tồn tại thì DWA sẽ thêm người mới vào trong nhóm này. Nếu chưa tồn tại thì DWA sẽ tạo nhóm mới và thêm người vào nhóm này. Kể cả nếu như chỉ có một người ở trong nhóm thì DWA vẫn cần cấp quyền truy cập cho người đó để vào được nhóm

- Ta cần cẩn trọng bới leavers và movers. Vấn đề ở đây là bộ phận HR có nhận thức về leaver nhưng họ lại không có thông tin tới DWA nên ta cần tổ chức một quá trình nơi mà HR thông báo tới BIM hay DWA khi họ nghỉ việc

- Một điều khác ta cần để ý đó là về yêu cầu được truy cập ngắn hạn cho một project ngắn ngày => Dễ hiểu là khi mà dự án kết thúc thì ta cần bỏ đi quyền truy cập của họ

## Managing database

- Quản lý data warehouse database là một quá trình duy trì database được sử dụng bởi data warehouse. Duy trì một database có nghĩa là đảm bảo dung lượng ổ đĩa đủ lớn luôn khả dụng, backups được diễn ra thường xuyên, chỉ mục được duy trì và tối ưu, thống kê database luôn được cập nhật

- Ở trong SQL Server-based data warehouse system ta có stage database, ODS hay NDS database, DDS database(s) và Analysis Services database (cubes)

- Vậy vì sao duy trì database trong data warehouse lại quan trọng lại vậy? Bởi hầu hết data trong data warehouse được lưu trữ ở trong database và data trong database được sử dụng cho việc hỗ trợ đưa ra quyết định về business

- Phạm vi của việc quản lý data warehouse database bao gồm những công việc sau:

* Maintaining capacity
* Backing up database
* Testing database restore
* Maintaining indexes
* Maintaining table partitions
* Changing the schema
* Changing the data
* Creating a DDS database

## Making schema changes

- Sau khi data warehouse goes live và vận hành thì thi thoảng sẽ có sự thay đổi trong các quá trình nghiệp vụ, doanh nghiệp khi chúng sinh ra thêm các data mới hay thay đổi hay cắt giảm data

- Ta không nên đánh giá thấp size hay là độ phức tạp của công việc này. Mang một phần mới của data dến data warehouse yêu cầu data modeling, schema changes, ETL modifications, new data quality rules, ETL testing, integration testing và rất nhiều công đoạn kiểm thử hồi quy khác

## Updating applications

- Sau khi data warehouse đã ở trong production, sẽ có một vài yêu cầu từ users muốn nâng cao, nâng cấp các ứng dụng như thêm cột ở trong một báo cáo, thêm một thuốc tính trong cube hay là update tính toán trong BI application => Sau khi data được modify thì ứng dụng sẽ được update

- Trong yêu cầu, user muốn cụ thể area nào của data warehouse cần thay đổi, thay đổi đó là gì, tổng hợp củ chức năng mới và screen/page mock-up nếu như nó yêu cầu sự thay đổi về giao diện người dùng. Nếu đó là báo cáo thay đổi, yêu cầu cần đề cập đến những thay đổi dữ liệu nào được yêu cầu, cũng như bất kỳ thay đổi nào đối với bố cục báo cáo. Điều này là cần thiết để bạn hiểu tác động của sự thay đổi đối với các ứng dụng kho dữ liệu

- Một yêu cầu thay đổi ứng dụng sẽ không yêu cầu sự thay đổi về lược đồ. Data luôn sẵn sàng ở trong data warehouse nhưng không được đưa đến application

## Summary

- Ta đã trao đổi về việc quản trị data warehouse, bao gồm việc giám sát ETL, quản tị database, quản lý security, mang data mới và cập nhật ứng dụng