KHOA CÔNG NGHỆ THÔNG TIN-ĐHKHTN
MÔN HỌC

Chapter 4
ETL PROCESS

Giáo viên: Hồ Thị Hoàng Vy

TPHCM, 8-2021

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

Goals

- After complete this chapter, student will:
  - Define the ETL approach and architecture
  - Able to extract, transform and load data into a datawarehouse

Reference

- Vincent Rainardi - Building a Data Warehouse: With Examples in SQL Server

Main topic

- **Introduction to ETL**
- ETL approach and architecture
- Extracting data
- Populating the DW

Introduction to ETL

- ETL is the process of retrieving and transforming data from the source system and putting it into the data warehouse.
- The most underestimated and time-consuming process in DW development
  - Often, 80% of development time is spent on ETL
- The ETL system is the foundation of the DW/BI project → its success makes or breaks the data warehouse.

Introduction to ETL

- ETL stands for Extract, Transform, and Load
  - **E**xtract
    - Extract relevant data from source system
  - **T**ransform
    - Transform data to DW format; deduce new data values, validate data checks
    - Data cleansing
    - …
  - **L**oad
    - loads the data into the DW

ETL - **Fundamental principles**

- Not to slow the source system down too much
- Be careful not to disturb the source system too much
- **The extraction to be as fast as possible**:
  - **Time**: such as five minutes if we can, not three hours
  - **Size**: as small as possible, such as 10MB per day, not 1GB per day.
  - **Frequency**: once a day if we can, not every five minutes

ETL - **Fundamental principles**

- The change in the source systems to be as minimal as possible
  - Should not creating triggers for capturing data changes in every single table.
- Should not have any **leakage**
- Can **recover** without data loss or damage

Main topic

- Introduction to ETL
- **ETL approach and architecture**
- Extracting data
- Populating the DW

ETL approach and architecture

- **Traditionall approach:**
  - To stage on disks or do transformation in memory
- **Alternative approaches based on**:
  1. where to perform the transformations
  2. who moves the data out of the source system
  3. Where to put ETL processes

ETL approach and architecture

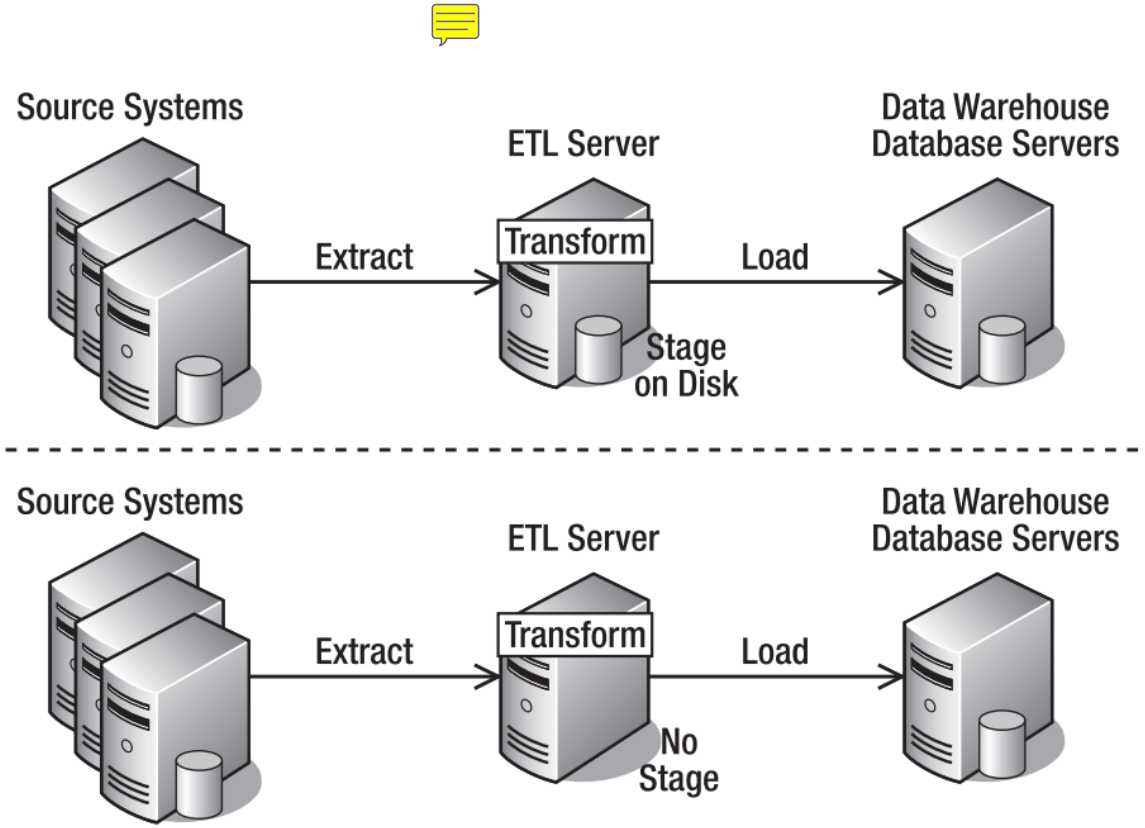
- **Traditional approach:**
  - Source system → ETL server (stage) → DW
  - Source system → ETL server (no staging, in memory) → DW

- The staging area is a physical database or files.
- Putting the data into the staging area means inserting it into the database or writing it in files.

ETL approach and architecture

Traditional approach:

- Transforming the data in memory is faster than putting it on disk first.
  - Small data?
  - Big data?



ETL approach and architecture

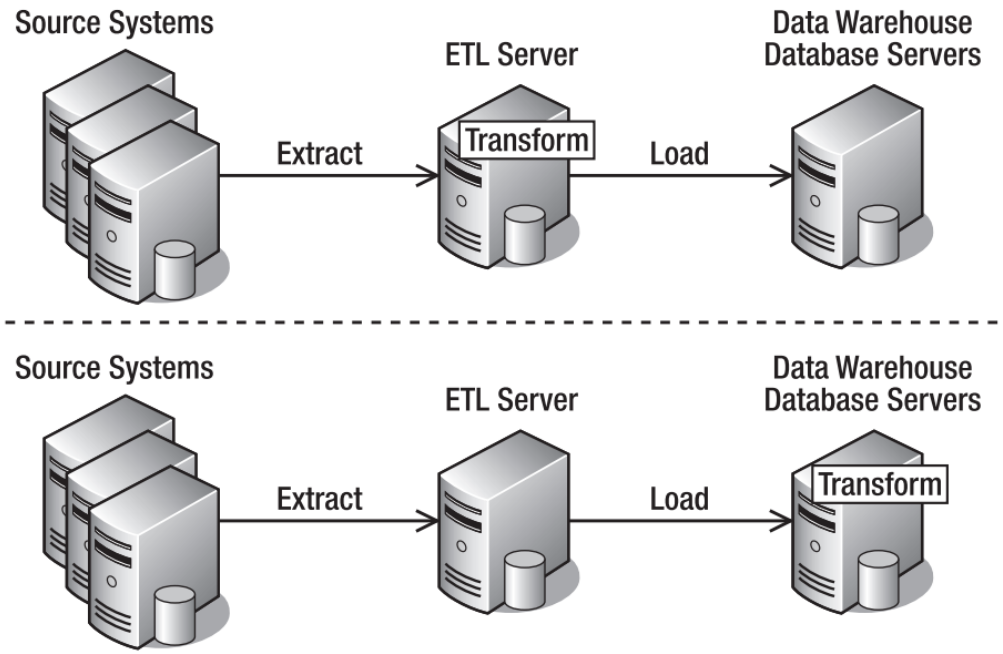
- **Traditionall approach:**
  - To stage on disks or do transformation in memory
- **Alternative approaches based on**:

1. **where to perform the transformations**
2. who moves the data out of the source system
3. Where to put ETL processes

ETL approach and architecture

- **E-L-T**: (Extract – Load – Transform): copy the source system (OLTP) data into the data warehouse and transform it there:
  - Pull data from source system
  - Load it into DW
  - Apply the transformation by updating data in DW

ETL approach and architecture



Strong ETL server
Strong software

strong DW database system - Usually Massively parallel processing (MPP)

ETL approach and architecture

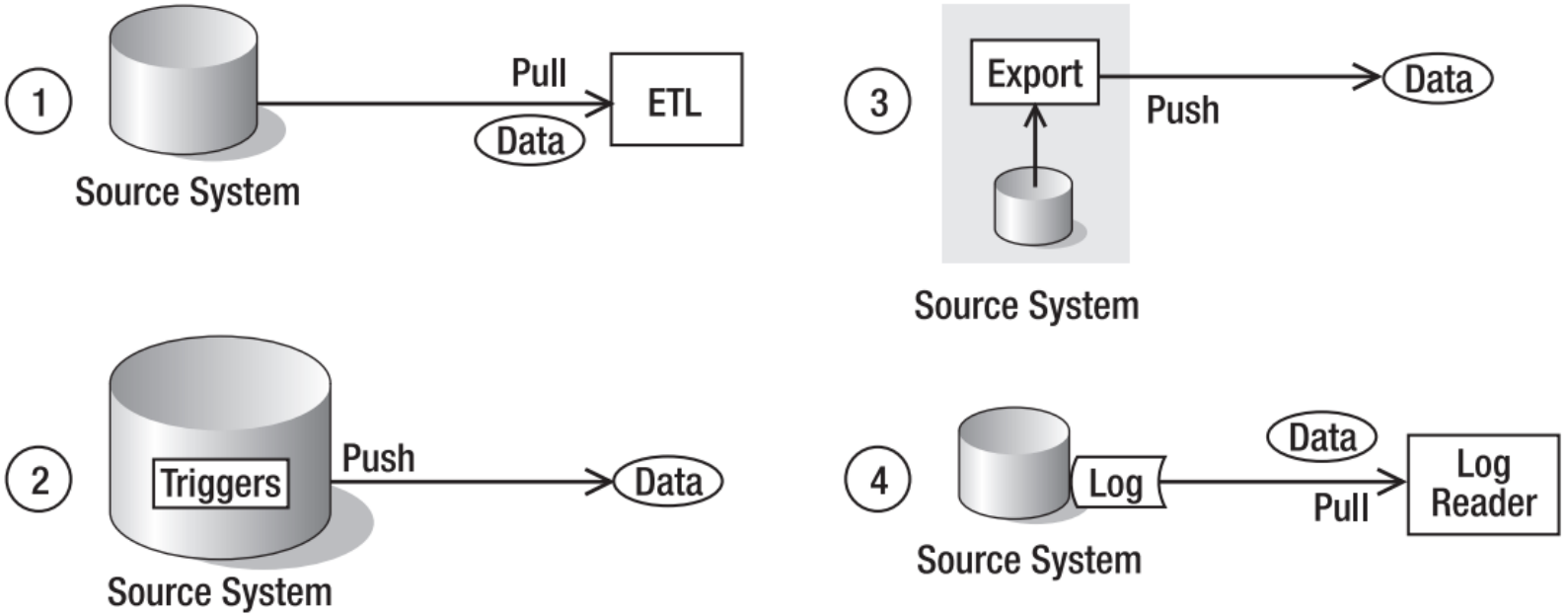
- **Traditionall approach:**
  - To stage on disks or do transformation in memory
- **Alternative approaches based on**:
  1. where to perform the transformations
  2. **who moves the data out of the source system**
  3. Where to put ETL processes

ETL approach and architecture

1. Pulls the data out by ***querying*** the source system database regularly
2. Implement a ***trigger*** in the source system database to push the data changes out
3. Install a ***schedule*** process within the source system to extract data periodically.
4. Implement a procedure to ***read*** the database ***logfiles*** of the DB source to discover every data change of the DB

ETL approach and architecture

- 4 method to execute the ETL process based on who moves the data out of the source system



ETL approach and architecture

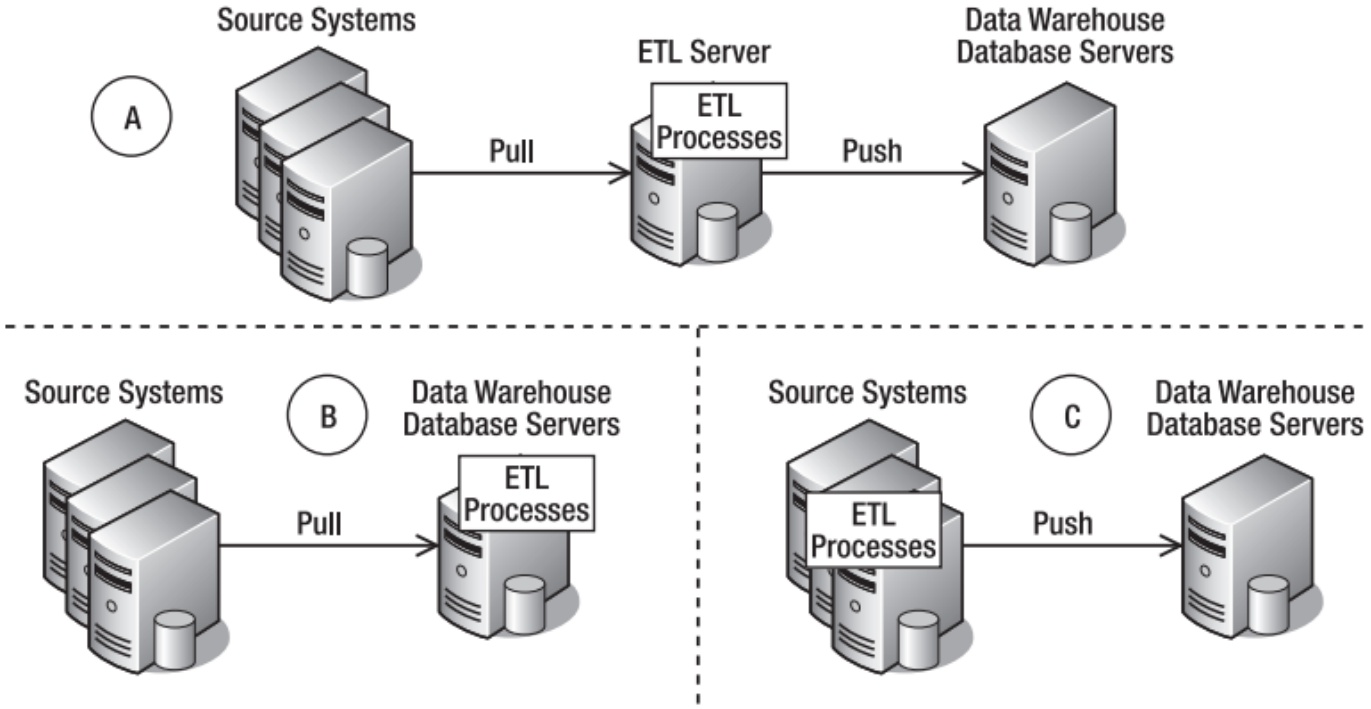
- **Traditionall approach:**
  - To stage on disks or do transformation in memory
- **Alternative approaches based on**:
  1. where to perform the transformations
  2. who moves the data out of the source system
  3. **Where to put ETL processes**

ETL approach and architecture

- **Three approaches based on where to put ETL processes**:
  - Execute the ETL processes in a separate ETL server that sits between the source system and the data warehouse server
  - Execute the ETL processes in the data warehouse server
  - Execute the ETL processes in the server that hosts the source system

ETL approach and architecture

- Three approaches based on where to put ETL processes:



Main topic

- Introduction to ETL
- ETL approach and architecture
- **Extracting data**
- Populating the DW

Extracting database

1. **Flat file**
2. Relational database
3. Others

Extraction from Flat files

- Example of flat file:

```
001|Nguyễn Ngọc Thảo|Accounting
002|Trần Thanh Toàn|Admin
003|Lê Thanh Nhi|Technical support
```

- Use bulk-insert in SQL command to load data from flat file
- Quick I/O, provide the best performance
  - Bulk insert table1 from 'file1' with (field terminator = '|')

Extraction from Flat files

- **Some remarks:**
  - Must know the flat file structure:
    - Field name
    - Delimiter
    - Type and length are not fixed
    - File name
  - Have access to the agreed-upon location (permission to delete/read files)
  - ….

Extracting database

1. **Flat file**
2. Relational database
3. Others

Extracting relational database

- **Three methods**:
  1. **Incremental extract**
  2. Fixed range
  3. Whole table every time

Incremental extract

- Download only the changed rows from the source system, not the whole table
  - New row
  - Newly deleted row
  - Newly updated row

- **Based on**: **timestamp** columns, identity columns, transaction dates, **triggers**, or a combination of them

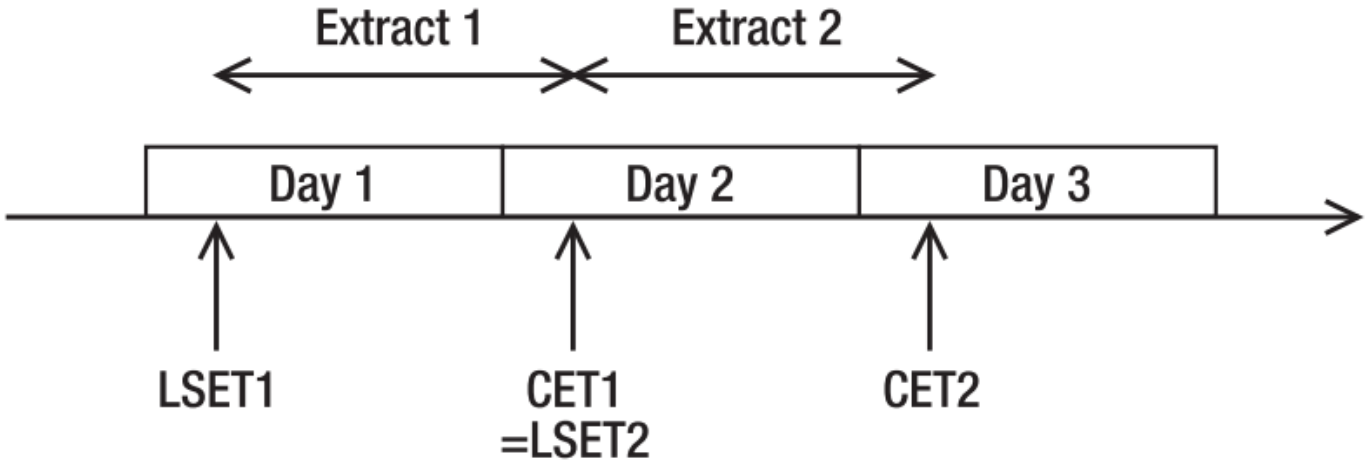
Incremental extract

- Using a “created”/last timestamp column, updated/order date/incremental orderID
  - check whether the timestamp columns are reliable
    - contain dummy values such as 1900-01-01, Blank, null, Last updated < created date….
- Every time the row in the table changes (insert/update), the timestamp is updated

Order ID	Order Date	Some Columns	Order Status	Created	Last Updated
45433	10/10/2007	Some Data	Dispatched	10/11/2007 10:05:44	10/12/2007 11:23:41
45434	10/15/2007	Some Data	Open	10/16/2007 14:10:00	10/17/2007 15:29:02
45435	10/16/2007	Some Data	Canceled	10/16/2007 11:23:55	10/17/2007 16:19:03
…					

Incremental extract

- Incremental extraction logic using **LSET** and **CET**
  - LSET: the time when data was last extracted.
  - CET is the time the **ETL package** started, **not** when the current **task started**



Incremental extract

Procedure to extract

1. Retrieve the LSET from the metadata database
2. Get the CET, which is passed in by the top-level ETL package
3. Extract the data
    1. **select** * **from** order_header **where** (created >= LSET and created < CET) or (last_updated >= LSET and last_update < CET)
4. Update meta data: writing CET as the new LSET value.

Incremental extract

- Business rule:
  - the order happened last week but just entered into the system today (pastdated orders).
  - If we apply the previous logic to the order_datecolumn, we will miss past-dated orders.
  - if you try to put the order date as 29 days ago → generate an error message.
  - ***select*** * ***from*** *order_header* ***where*** *order_date >= (LSET – 28 days) and created < CET*

Incremental extract

Another way of doing incremental extract is to use the order ID:

- Retrieve the Last Successfully Extracted ID (LSEI) from the metadata database.
- Select max(order_id) from order_header
- Set CEI = max
- Select * from order_header where order_id >= LSEI and order_id < CEI.
- Update meta data: LSEI = CEI.

Remarks

- Incremental Extract logic is fault tolerant
- if ETL doesn't run or it failed to run, the return result will:
  - no risk of missing the data
  - Not loading data that we loaded earlier

- **How about deletion?**
  - How do we know which orders have been deleted?

Discussion

- How about deletion?
  - soft delete, don't physically delete the record in the table
  - physically deleted
    - comparing the PK between the source table and the warehouse table
    - Using deletion trigger
      - A trigger is the most reliable approach in ETL
      - Can create separate triggers for delete, update, and insert
      - **Drawback**:

Fixed range

- Periodically extract a certain number of records or a certain period of time based on business constraint.
  - no reliable incremental identity column
  - no timestamp columns
  - timestamp columns are not reliable
  - …..
- EX: by the end of each month, extract the data of that month

Whole table

When:

- Deals with tables of small size
- No timestamp or identity column
- Neither incremental attribute
- No business constraint

Extracting database

1. Flat file
2. Relational database
3. **Others**

Extracting Other Source Types

- XML
- spreadsheet files (Excel)
- Web logs
- Binary file
- Webservice
- Emails
- ……

Main topic

- Introduction to ETL
- ETL approach and architecture
- Extracting data
- **Populating the DW**

GROUP DISCUSSION

- Design NDS + DDS database for Atremendous Entertainment Case study.
- Time: 30 minutes (from 12:55pm to 13:25PM)

Populating the DW

- Loading the stage
- Creating the data firewall
- Populating a normalized data store (NDS)
- Populating dimension data store (DDS)
  - Populating dimension tables
  - Populating fact tables

Stage loading (source → stage → NDS → DDS)

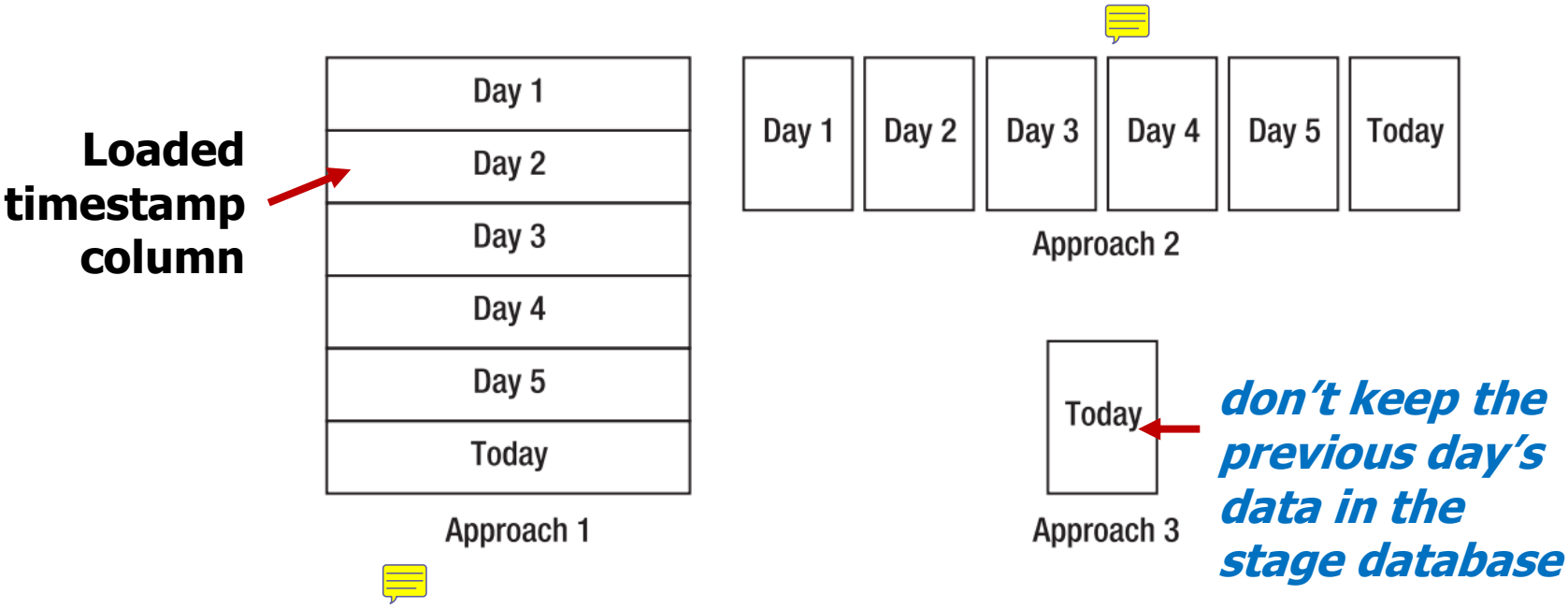
- Load the source system data into the stage
- Extract the data as soon as possible without doing too much transformation
  - the structure of the stage tables is similar to the source system tables
- It is better not to put any indexes or constraints in the stage database
  - to capture and report the “bad data”in the data quality process

Stage loading

- Three different approaches of how stage tables are structured:
  1. Keeps the previous day's data in the same table
  2. Keeps each day in a separate table
  3. Uses Just one table and truncate the table every time before loading

Stage loading

- Three different approaches of how stage tables are structured:



Data firewall

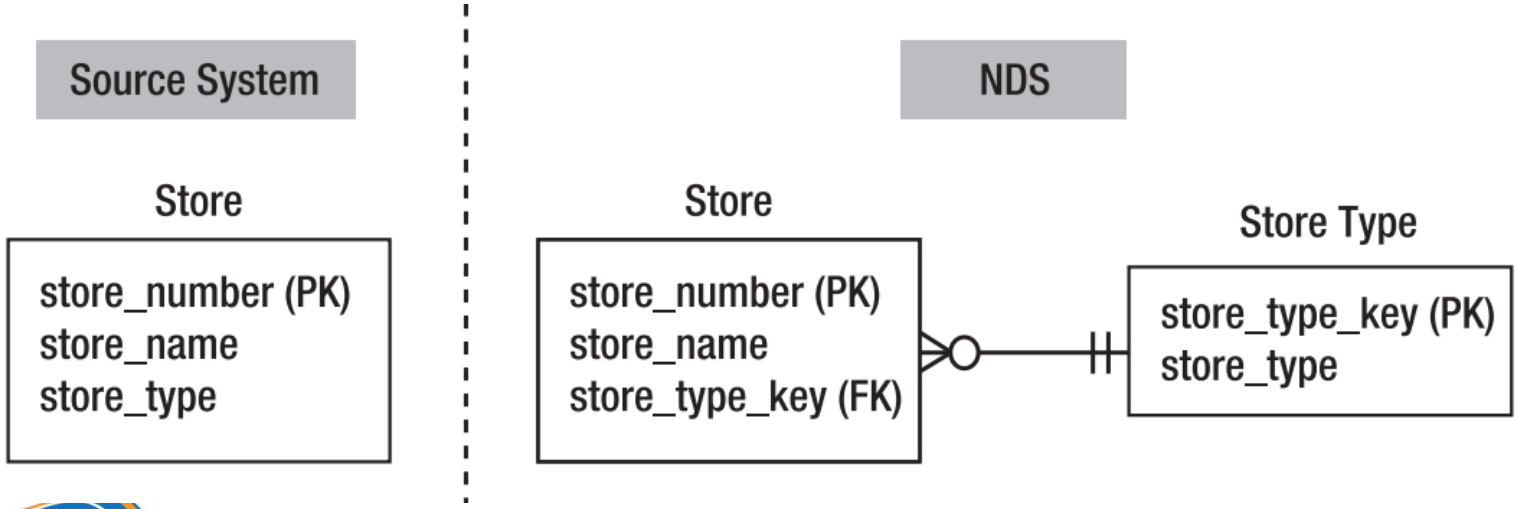
- The data firewall is a program that checks the incoming data, similar to the firewall concept in networking → ensures data quality
  - Physically, it is an SSIS package or a stored procedure
  - Place a data firewall between the stage and the
    - **reject** the data (not load it into the DW),
    - **allow** the data (load it into the DW)
    - **fix** the data (correct the data before loading it into the DW)

Populating NDS (source→ stage→ NDS→ DDS)

- Extract data then load it into the NDS database:
  - From the stage table
  - From the source system
- **Some remarks**
  - In the NDS, the tables are normalized
  - External data
  - Data conversion
  - Key management
  - Consider insert/ update issues

Normalization

- Normalization is a process of removing data redundancy by implementing normalization rules (1NF, 2NF, 3 NF, BCF…)
- A normalized data store is usually in third normal form or higher



EX

Store

store_number	store_name	store_type
1805	Perth	Online
3409	Frankfurt	Full Outlet
1014	Strasbourg	Mini Outlet
2236	Leeds	Full Outlet
1808	Los Angeles	Full Outlet
2903	Delhi	Online

Normalized

N_Store

store_number	store_name	store_type_key
1805	Perth	1
3409	Frankfurt	2
1014	Strasbourg	3
2236	Leeds	2
1808	Los Angeles	2
2903	Delhi	1

Store_Type

store_type_key	store_type
1	Online
2	Full Outlet
3	Mini Outlet

Each store has one store type
Each store type relates to zero or many stores

Normalization

- Data from source

store_number	store_name	store_type
2009	Dallas	Online
2237	London	Full Outlet
2014	San Francisco	Distribution Center

not exists

Store_NDS

store_number	store_name	store_type_key
1805	Perth	1
3409	Frankfurt	2
1014	Strasbourg	3

StoreType_NDS

store_type_key	store_type
1	Online
2	Full Outlet
3	Mini Outlet

store_type_key	store_type
1	Online
2	Full Outlet
3	Mini Outlet
4	Distribution Center

Store_NDS

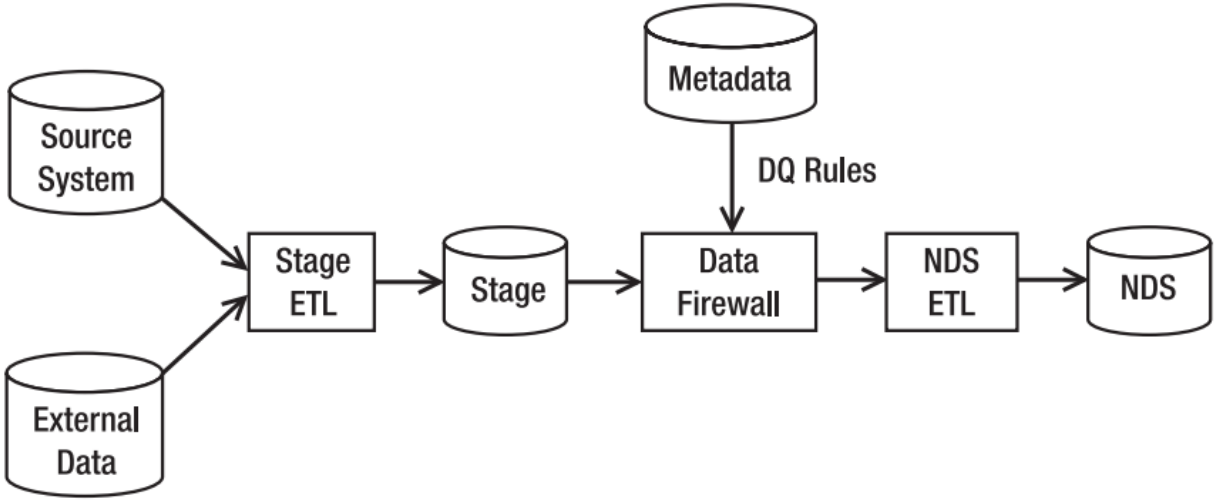
store_number	store_name	store_type_key
1805	Perth	1
3409	Frankfurt	2
1014	Strasbourg	3
2009	Dallas	1
2237	London	2
2014	San Francisco	4

External data

- Customer from Source system: Congo (Zaire)
- Customer from external data: Congo

➔ should we ***create*** “Congo (Zaire)” as a new row?

Or ***replace*** “Congo (Zaire)” with “Congo”?



External data

- **TIPS: to prevent the ETL from creating duplicate entries in the NDS country table**
  - Create a rule in the data quality routine to replace “Congo (Zaire)”
  - Then the NDS ETL looks up “Congo” in the NDS country table

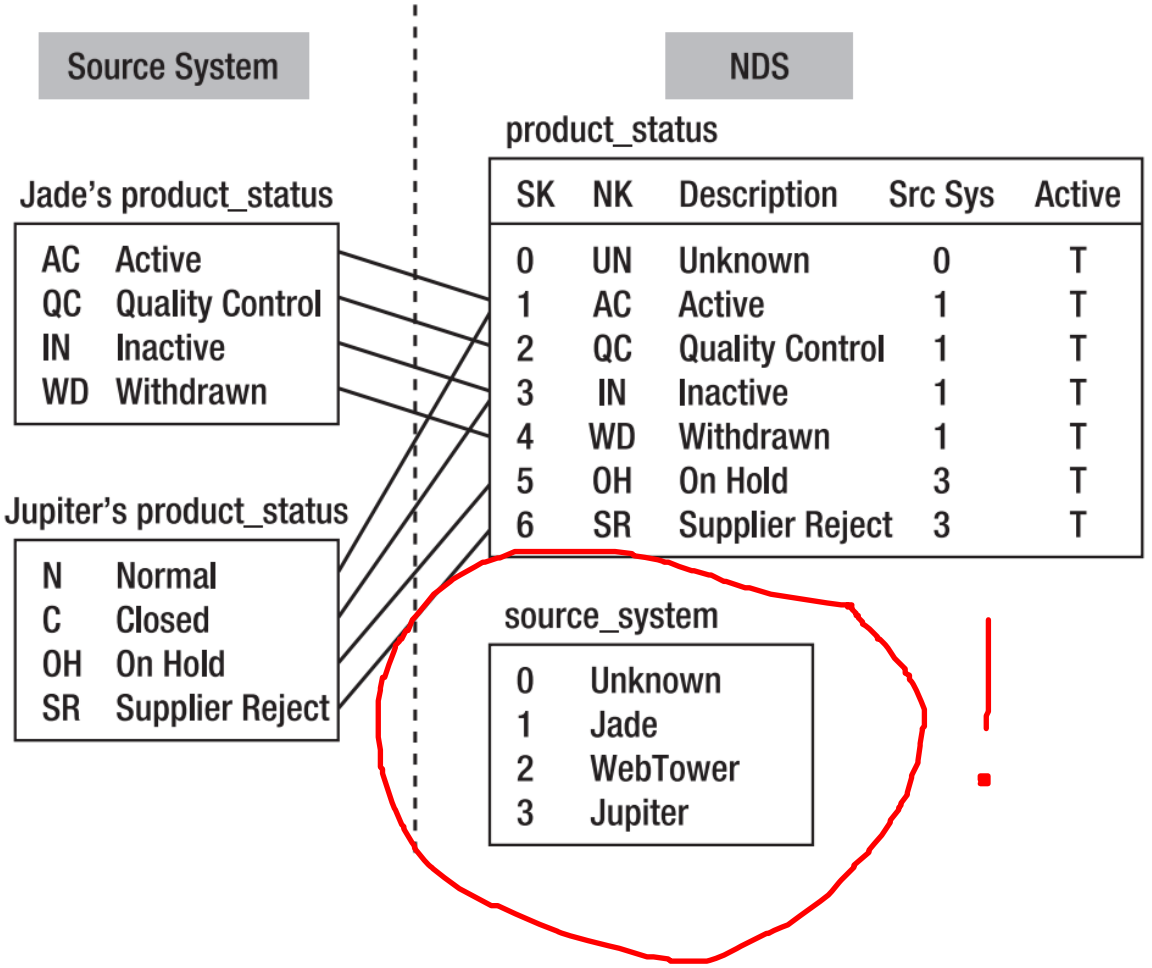
Key management

VỀ NHÀ ĐỌC !

- To be able to adapt the key changes in the source system(s).
- The data warehouse key is known as a surrogate key (SK) → enables the integration of several source systems
- The source system key is known as a natural key (NK)

Key management

- The mapping for the product_status table between the source systems and the NDS



Trong NDS có thêm bảng này

Populating DDS - Dimension Tables

- DDS tables: **fact** tables and **dimension** tables
- The dimension tables in the DDS are ***denormalized***
- Like populating NDS, do an **UPSERT** operation to update or insert the source row depending on whether it exists in the target
- ***Several issues need to be considered:***
  - Incremental loading,
  - Key management,
  - Denormalization,
  - Slowly changing dimension (SCD)

Populating DDS - Dimension Tables

- The dimension tables in the DDS are ***denormalized***
- Like populating NDS, do an **UPSERT** operation to update or insert the source row depending on whether it exists in the target
- ***Several issues need to be considered:***
  - **Incremental loading,**
  - Key management,
  - Denormalization,
  - Slowly changing dimension (SCD)

In the NDS everything is timestamped → load only the NDS rows that changed since the last ETL run

Populating DDS - Dimension Tables

- The dimension tables in the DDS are ***denormalized***
- Like populating NDS, do an **UPSERT** operation to update or insert the source row depending on whether it exists in the target
- ***Several issues need to be considered:***
  - Incremental loading,
  - **Key management,**
  - Denormalization,
  - Slowly changing dimension (SCD)

The Surrogate key (SK) are managed in the NDS ➔ can guarantee that all DDSs use the same SK.

Populating DDS - Dimension Tables

- The dimension tables in the DDS are ***denormalized***
- Like populating NDS, do an **UPSERT** operation to update or insert the source row depending on whether it exists in the target
- ***Several issues need to be considered:***
  - Incremental loading,
  - Key management,
  - **Denormalization,**
  - Slowly changing dimension (SCD)

To load the store dimension in the DDS, in the NDS we need to join the storetable with the store_type table

Populating DDS - Dimension Tables

- The dimension tables in the DDS are ***denormalized***
- Like populating NDS, do an **UPSERT** operation to update or insert the source row depending on whether it exists in the target
- ***Several issues need to be considered:***
  - Incremental loading,
  - Key management,
  - Denormalization,
  - **Slowly changing dimension (SCD)**

SCD type 1(overwrite),
SCD type 2 (rows),
SCD type 3 (column)

Populating DDS - Fact Tables

- Fact tables are normally large tables
- Table partitioning can speed up the update operation significantly when correctly applied to a fact table
- The dimensional key index is required to find the row that we want to update

Assuring Data Quality

- when building a DW, it is important to think about data quality **as early as possible**
- The data quality process includes the activities to make sure the data in the DW is correct and complete
  - set up rules that define what “bad data”
  - Reporting
  - Monitoring
  - Cleaning/ Correcting

Assuring Data Quality

- Example: customers can purchase a product, or they can subscribe to a package
  - first subscription date > last cancellation date → invalid condition
  - Question:
    - The last cancellation date is wrong
    - The first subscription date is wrong
    - or both??

Assuring Data Quality

- Data Cleansing and Matching
- Cross-checking with External Sources
- Data Quality Rules

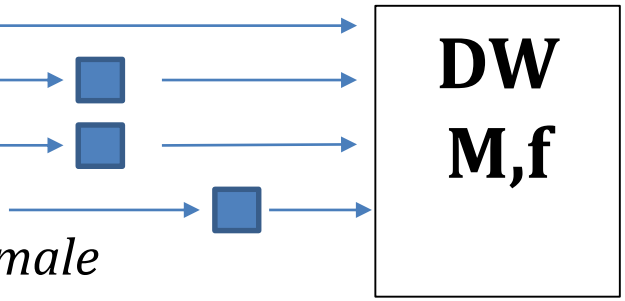
Data Cleansing and Matching

- **Data cleansing**, or data scrubbing is the process of *identifying* and *correcting* **dirty data**
  - Dirty data means incomplete, wrong, duplicate, or out-of-date data
- **Ex**:
  - Checking stores' tables to make sure the store names, store numbers, store types, and store addresses are all correct
  - Making sure that there are no duplicate customer records → **data matching**
    - **use = for matching numeric data. For example, "if A = B, then…."**
    - **is 5.029 the same as 5.03?**
    - **03/01/2008 the same as 01/03/2008?**

Data Cleansing and Matching

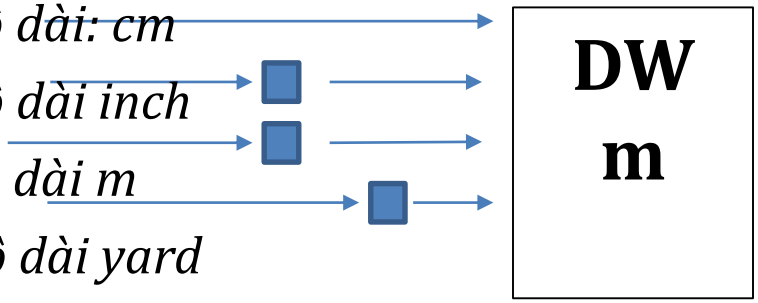
Different unit of gender

- *Application A: - m,f*
- *Application B: - yes, no*
- *Application C: - 1, 0*
- *Application D: - male, female*



Different unit of measure

- *Application A: - độ dài: cm*
- *Application B: - độ dài inch*
- *Application C: - độ dài m*
- *Application D: - độ dài yard*



***Chuyển đổi khác**: đơn vị tiền tệ, các thông tin tính toán, kiểu dữ liệu, độ dài dữ liệu…*

Data Cleansing and Matching

- When consolidating data from different source systems, it is possible that a piece of data is available in one system but is not in the other system.
- **Ex**:
  - address fields of system A (address1, address2, city, county, ZIP, and country)
  - address fields of system B (address1, city, county, ZIP)
  - ➔ system B does not have the ***address2*** field and the ***country field***
  - ➔ ***solution?***

Data Cleansing and Matching

- Sometimes the same data may contain different things:
  - In system A,
    - a column called “Total Order Value” may contain taxes, discounts, credit card charges, and delivery charges
    - term weekly traffic may refer to **unique** web site visitors
  - In system B
    - Column “Total Order Value” does not contain delivery charges
    - term weekly traffic means nonunique web site visitors
  - **Solution?**

SQL server matching

- Three types of matching logic:
  - **Exact**: all characters are the same, for example “Los Angeles” and “Los Angeles.”
  - ➔ using a **Lookup** transformation
  - **fuzzy** (approximate): finds how similar a set of data is to another set of data
    - using the Fuzzy Lookup
    - “You can’t hurry love” and “You cannot hurry love” have a similarity score of 0.81666672 and a confidence level of 0.59414238
  - **rule based**: use certain rules and data to identify a match
    - In product names “movie”is the same as “film”. → implemented with Script Component
    - “For product code, omit the spaces when comparing”so that “KL 7923 M” is the same as “KL7923M.”

Meta data

- In general, Metadata means data that describes data
- In data warehousing, metadata contains :
  - The definitions of data (the meaning and source of each column),
  - The definition of the data warehouse itself (the data store structure,the ETL processes,and the data quality),
  - The definition of the related systems (ex: 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)

Meta data

- **7 kinds of metadata**:
  - **Data definition and mapping metadata** contains the meaning of each fact and dimension column and where the data is coming from.
  - **Data structure metadata** describes the structure of the tables in each data store.
  - **Source system metadata** describes the data structure of source system databases.
  - **ETL process metadata** describes each data flow in the ETL processes.
  - **Data quality metadata** describes data quality rules, their risk levels, and their actions.
  - **Audit metadata** contains a record of processes and activities in the data warehouse.
  - **Usage metadata** contains an event log of application usage.

Data definition and mapping metadata

- Contains the meaning of each fact and dimension column and where the data is coming from.

table_key	column_key	description	sample_value	source_column_key
56	112	The surrogate key of the product dimension. It is unique, is not null, and is the primary key of the product dimension.	3746	88
56	113	Natural key. Product code is the identifier and primary key of the product table in Jupiter. It is in AAA999999 format.	FGA334288	89
56	114	The product name.	The Panama Story DVD	90
56	115	The product description.	The Panama Story movie on DVD format	91
56	116	The song/film/book title.	The Panama Story	92

```
create table data_definition
(column_key int not null
,table_key int not null
,column_type_key int not null
,description varchar(200)
,sample_values varchar(50)
,source_column_key int not null
,create_timestamp datetime not null
,update_timestamp datetime not null
,constraint pk_data_definition
primary key clustered (column_key)
) go
create index data_definition_description
ondata_definition(description)
go
```

Data definition and mapping metadata

Column Types table

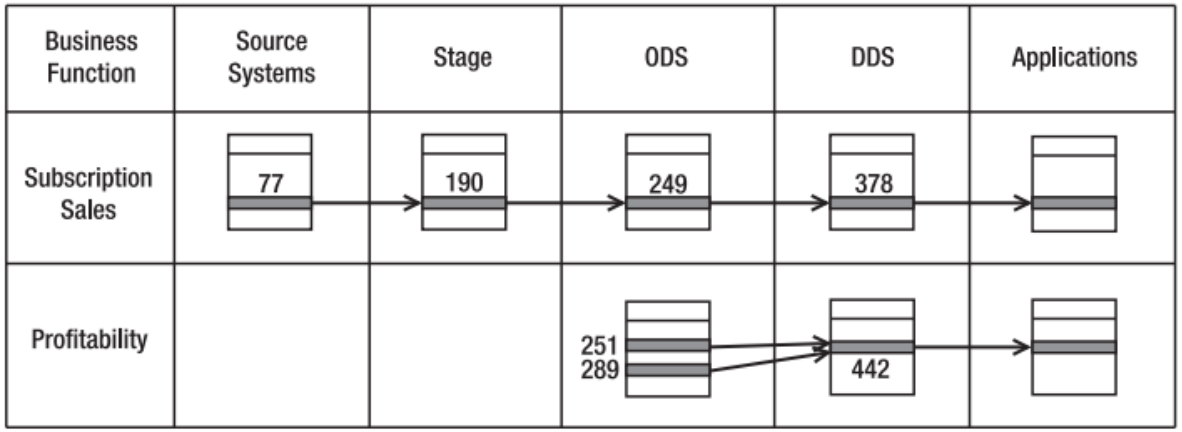
- Columns in data warehouse tables serve different purposes.
- A column type indicates these purposes.

Column Type	Location	Description
Surrogate key	DDS dimension tables	A single not null column that uniquely identifies a row in a dimension table.
Natural key	DDS dimension tables	Uniquely identifies a dimension row in the source system.
Dimensional attribute	DDS dimension tables	Describes a particular property of a dimension.
Degenerate dimension	DDS fact tables	Identifies a transaction in the source system. A natural key of a dimension without any attributes.
SCD support	DDS dimension tables	Columns that support slowly changing dimension such as `is_active`, `effective_date`, and `expiry_date`.
Measure	DDS fact tables	Columns in the fact table that contain business measurements or transaction values.
Fact key	DDS fact tables	A single not null column that uniquely identifies a row on a fact table.
System	All data stores	Auxiliary columns created by the system for system usage such as `create_timestamp` and `update_timestamp`.
Transaction	ODS and NDS tables	Column in normalized tables containing business transaction values, such as order tables.
Master	ODS and NDS tables	Columns in normalized tables that contain master data such as stores, products, and campaigns.
Stage	Stage tables	Columns in stage tables containing business data.

```
Create table data_mapping
(data_mapping_key int not null
identity(1,1)
,column_key int not null
,source_column_key int
,create_timestamp datetime not null
,update_timestamp datetime not null
,constraint pk_data_mapping
primary key clustered
(data_mapping_key)
) go
```

Data definition and mapping metadata

- The data flow diagram between data stores in a data warehouse.



- Data Mapping Table for the above Data Flow

data_mapping_key	column_key	source_column_key	create_timestamp	update_timestamp
1	378	249	2007-10-24 09:23:48	2007-11-18 14:10:08
2	249	190	2007-10-24 09:28:36	2007-11-19 11:05:16
3	190	77	2007-10-24 09:31:13	2007-10-24 09:31:13
4	442	251	2007-11-04 17:01:55	2007-12-18 15:09:42
5	442	289	2007-11-04 17:03:29	2007-11-04 17:03:29

Data structure metadata

- **Data structure metadata** describes the structure of the tables in each data store.

Data Structure Metadata Tables

Table Name	Description
ds_data_store	Lists all data stores in the data warehouse
ds_table	Lists tables in each data store
ds_table_type	Lists table types such as dimension table, fact table, and so on
ds_column	Lists all columns in each table
ds_column_type	Lists column types such as the surrogate key column, measure, and so on

Data Structure Metadata:ds_data_storeTable

key	data_store	description	collation	current_ size	growth
1	Stage	Staging area	SQL_Latin1_General_CP1_CI_AS	70	10
2	NDS	Normalized data store	SQL_Latin1_General_CP1_CI_AS	125	25
3	DDS	Dimensional data store	SQL_Latin1_General_CP1_CI_AS	150	25
4	Meta	Metadata	SQL_Latin1_General_CP1_CI_AS	10	5

cdio 4.0

fit@hcmus

ETL process metadata

- Describes each data flow in the ETL processes.

ETL Processes Metadata:packageTable

key	name	description	schedule
…			
4	Stage daily incremental	This SSIS package extracts the following data from the source system incrementally and loads it onto the stage: customer, permission, product, store, product purchases, package subscription, and communication subscription.	Runs every day including weekends at 3 a.m.
5	Stage daily full reload	This SSIS package extracts the following data from the source system by downloading the whole table: customer type, customer status, interest, household income, occupation, currency, artist, product status, product type, product category, package, and package type.	Runs every day at 3 a.m.
11	NDS product	This loads product-related tables from the stage to the NDS, in other words, the ones with product, product status, product type, product category, package, and package type data.	Runs daily at 3 a.m.
12	NDS customer	This loads customer-related tables from the stage to the NDS, in other words, the ones with customer, customer status, customer type, address, address type, e-mail address, e-mail address type, permission, phone number, phone number type, household income, occupation, and interest data.	Runs daily at 3 a.m.
…			

ETL Processes Metadata:data_flowTable

key	name	description	source	target	transformation	package	status	LSET	CET
…									
8	stage_product	Extracts Jade and Jupiter product tables incrementally based on the last-updated date and puts them in the product stage table.	Jade.product, Jade.product_detail, Jupiter.prod_hd, Jupiter.prod_dt	stage.product	Adds 9000 on Jupiter product code; dedupe two source systems based on product group and product code.	4	1	11/27/2007 04:06:09	11/27/2007 04:06:09
9	nds_product	Loads product data from the stage to the NDS.	state.product	stage.product	Upsert and keying; DQ unit_cost with rule 29	11	1	11/27/2007 04:07:11	11/27/2007 04:07:11
…									

key	status
…	
0	Unknown
1	Success
2	Failed
3	In progress
…	

Status Table

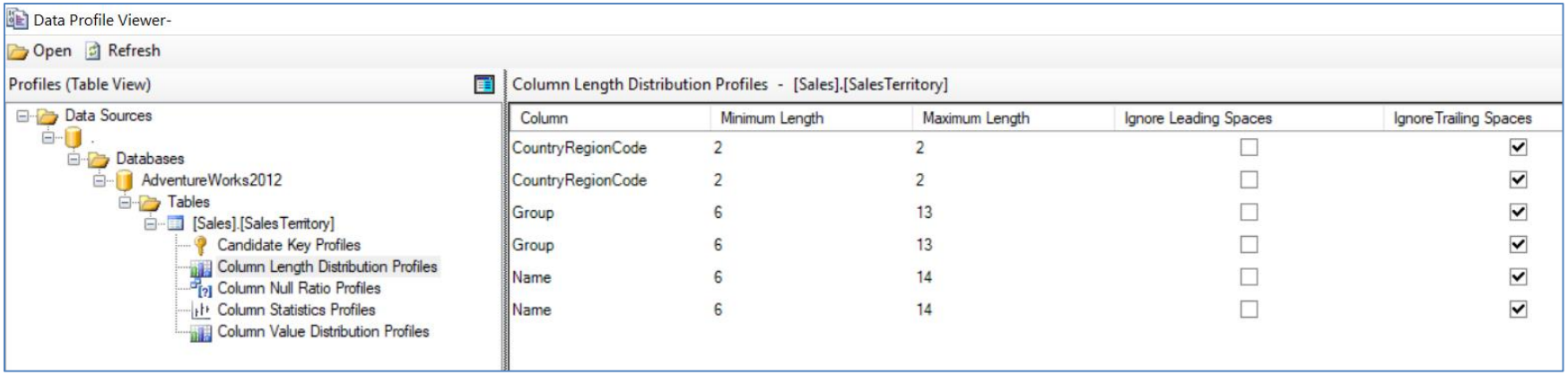
Audit Metadata

- Contains the results of every process and activity in the data warehouse, including data loading process (ETL), creation of specific purpose DDS (mart), manual data modification (for example, updating DQ user table), DQ rule validation,…

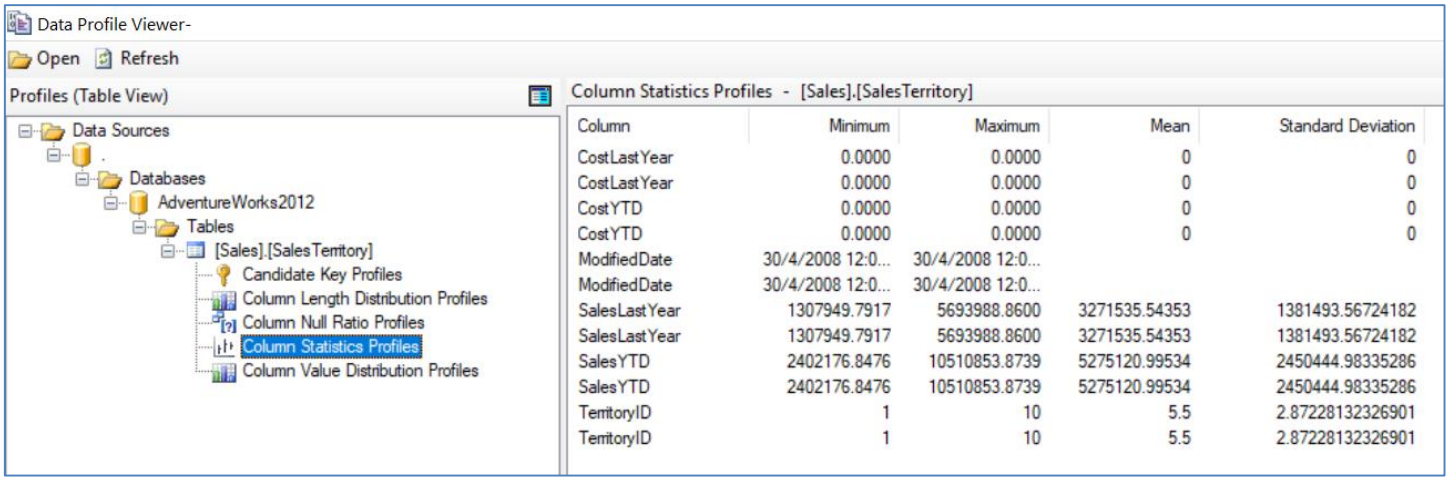
key	event_type	event_ category	timestamp	object	data_flow	rows	note
…							
12418	26	5	2007-11-24 03:04:09	56	12	23102	
12419	27	6	2007-11-24 03:05:19	87	23	23102	
12420	28	7	2007-11-24 03:06:11	112	38	23102	
12581	39	18	2007-11-25 14:36:06	29			Login successful
15013	59	21	2007-11-26 00:09:15	35			Applying patch 19
…							

Audit Metadata:event_logTable

Data profiling



Column	Minimum Length	Maximum Length	Ignore Leading Spaces	IgnoreTrailing Spaces
CountryRegionCode	2	2	☐	☑
CountryRegionCode	2	2	☐	☑
Group	6	13	☐	☑
Group	6	13	☐	☑
Name	6	14	☐	☑
Name	6	14	☐	☑



Column	Minimum	Maximum	Mean	Standard Deviation
CostLastYear	0.0000	0.0000	0	0
CostLastYear	0.0000	0.0000	0	0
CostYTD	0.0000	0.0000	0	0
CostYTD	0.0000	0.0000	0	0
ModifiedDate	30/4/2008 12:0...	30/4/2008 12:0...		
ModifiedDate	30/4/2008 12:0...	30/4/2008 12:0...		
SalesLastYear	1307949.7917	5693988.8600	3271535.54353	1381493.56724182
SalesLastYear	1307949.7917	5693988.8600	3271535.54353	1381493.56724182
SalesYTD	2402176.8476	10510853.8739	5275120.99534	2450444.98335286
SalesYTD	2402176.8476	10510853.8739	5275120.99534	2450444.98335286
TerritoryID	1	10	5.5	2.87228132326901
TerritoryID	1	10	5.5	2.87228132326901

Data profiling