LECTURE 13

Issues of ETL

Learning Goals

- Why ETL Issues?
- Understand Complexity of problem/work underestimated
- Multiple sources for same data element

Why ETL Issues?

Data from different source systems will be different, poorly documented and dirty. Lot of analysis required.

Easy to collate addresses and names? Not really. No address or name standards.

Use software for standardization. Very expensive, as any "standards" vary from country to country, not large enough market.

It is very difficult to extract the data from different source systems, because by definition they are heterogeneous, and as a consequence, there are multiple representations of the same data, have inconsistent data formats, have very poor documentation and have dirty data etc. Therefore, you have to figure different ways of solving this problem, requiring lot of analysis.

Consider the deceptively simple task of automating the collating of addresses and names from different operational databases. You can't predict or legislate format or content, therefore, there are no address standards, and there are no standard address and name cleaning software.

There is software for certain geographies, but the addresses and names in US will be completely different from that in Austria or Brazil. Every country finds a way of writing the addresses differently, that is always a big problem. Hence automation does not solve the problem, and also has a high price tag in case of 1st generation ETL tools ($200K- $400K range).

Why ETL Issues?

Things would have been simpler in the presence of operational systems, but that is not always the case.

Manual data collection and entry. Nothing wrong with that, but potential to introduces lots of problems.

Data is never perfect. The cost of perfection, extremely high vs. its value.

Most organizations have ERP systems (which are basically operational systems) from where they load their DWH. However, this is not a rule, but used in an environment where transaction processing systems are in place, and the data is generated in real time

However, in the case of a country-wide census, there are no source systems, so data is being collected directly and manually. People go form door to door and get forms filled, subsequently the data in its most raw form is manually entered into the database. There is nothing wrong with that, the problem being human involvement in data recording and storage is very expensive, time consuming and prone to errors. There have to be control and processes to collect as clean a data as possible.

People have a utopian view that they will collect perfect data. This is wrong. The cost of collecting perfect data far exceeds its value. Therefore, the objective should be to get as clean data as can be based on the given constraints.

"Some" Issues

- Usually, if not always underestimated
- Diversity in source systems and platforms
- Inconsistent data representations
- Complexity of transformations
- Rigidity and unavailability of legacy systems
- Volume of legacy data
- Web scrapping

ETL is something which is usually always underestimated; consequently there are many many

issues. There can be an entire course on ETL. Since this course is not only about ETL, therefore, we will limit our self to discussing only some of the issues. We will now discuss some of the more important issues one-by-one.

Complexity of problem/work underestimated

- Work seems to be deceptively simple.
- People start manually building the DWH.
- Programmers underestimate the task.
- Impressions could be deceiving.
- Traditional DBMS rules and concepts break down for very large heterogeneous historical databases.

At first glance, when data is moved from the legacy environment to the data warehouse environment, it appears there is nothing more going on than simple extraction of data from one place to the next. Because of the deceptive simplicity, many organizations start to build their data warehouse manually. The programmer looks at the movement of data from the old operational environment to the new data warehouse environment and declares "I can do that!" With pencil and writing pad in hand the programmer anxiously jumps into the creation of code in the first three minutes of the design and development of the data warehouse.

However, first impressions can be deceiving - in this case very deceiving. What at first appears to be nothing more than the movement of data from one place to another quickly turns into a large and complex task, actually a nightmare? The scope of the problem being far larger and far more complex than the programmer had ever imagined

Diversity in source systems and platforms

Platform	OS	DBMS	MIS/ERP
Main Frame	VMS	Oracle	SAP
Mini Computer	Unix	Informix	PeopleSoft
Desktop	Win NT	Access	JD Edwards
	DOS	Text file	

Dozens of source systems across organizations

Numerous source systems within an organization

Need specialist for each

Diversity in source systems and platforms

There are literally more than a dozen different source system technologies in use in the market today. Example of the kind of source system technologies that you often see are shown in Table 17.1. The team doing ETL processing in any given organization probably has to understand at least 6 of them. You will be surprised to find big banks and telecommunication companies that have every single one of these technologies; in a single company. It is a nightmare to understand them. The only viable solution is to have a technical expert that knows how to get data from every single one of these formats. In short it's a nightmare.

Same data, different representation

Date value representations

Examples:

970314 1997-03-14

03/14/1997 14-MAR-1997

March 14 1997 2450521.5 (Julian date format)

Gender value representations

Examples:

- Male/Female - M/F
- 0/1 - PM/PF

There are 127 different ways to spell AT&T; there are 1000 ways to spell duPont. How many ways are there to represent date? What is the Julian date format? All of these questions and many more are at the core of inconsistent data representation. Don't be surprised to find source system with Julian date format or dates stored as text strings.

Consider the case of gender representation. You might think it is just male and female. Wrong. It's not just male and female, it also includes unknown, because for some people you don't have their gender data. If this was not sufficient, there is another twist to the gender i.e. instead of M and F, you come across PM and PF. What is this? This is probable male and probable female i.e. based on the name the gender has been guessed. For example, if the name is Mussarat it is probable female. It is not for sure, but we think it is. If it's Riffat, it is probable male.

So you get all these weird and strange representations in operational data, while in a DWH there has to be only ONE and consistent representation. You just can't dump all the operational data independent of which source system it came from that has a different data representation. Remember in the data model you should have non-overlapping consistent domains for every attribute.

Multiple sources for same data element

- Need to rank source systems on a per data element basis.
- Take data element from source system with highest rank where element exists.
- "Guessing" gender from name
- Something is better than nothing?
- Must sometimes establish "group ranking" rules to maintain data integrity.
- First, middle and family name from two systems of different rank. People using middle name as first name.

This means you need to establish ranking in the source system on per element (attribute) basis. Ranking is all about selecting the "right" source system. Rank establishment has to be based on which source system is known to have the cleanest data for a particular attribute. Obviously you take the data element from the source system with the highest rank where the element exists. However, you have to be clever about how you use the rank.

For example, consider the case of the gender data coming from two different source systems A and B. It may be the case that the highest quality data is from source system A, where the boxes for the gender were checked by the customers themselves. But what if someone did not check the gender box? Then you go on to the next cleanest source system i.e. B, where the gender was guessed based on the name.

Obviously the quality of available data for source system B is not as good as that of source system A, but since you do not have data for this particular individual in source systems A, so it is better to have something then nothing. This is arguable i.e. maybe it is better to have nothing than to have dirty data. The point is if you have some level of confidence in the data you should take it. Typically it is a good idea to remember from where you took the data, so you can use this information from an analytic point of view e.g. where you get clean data etc.

Consider the case of name i.e. first name, middle name and family name and two source systems i.e. C and D. Assume that source system C has higher quality data entry, and management and processes and controls as compared to D. Also assume that source system C does not have the middle name, while source system D has the complete name. Since C has a higher precedence, so you take the first name and the family name from it. But you really would like to have the middle name so you take it from source system D. This turns out to be a big problem, because people sometimes use their middle name as their first name e.g. *Imran Baqai* instead of *Shezad Imran Baqai.*

Complexity of required transformations

Simple one-to-one scalar transformations

- 0/1 ? M/F

One-to-many element transformations

- 4 x 20 address field ? House/Flat, Road/Street, Area/Sector, City.

Many-to-many element transformations

- House-holding (who live together) and individualization (who are same) and same lands.

There is a spectrum of simple all the way up to very complex transformations that you can

implement. And you probably end up with many in most DWH deployments. The transformations are typically divided into three categories as follows:

? Simple one-to-one scalar transformations.
? One-to-many element transformations.
? Complex many-to-many element transformations.

Simple scalar transformation is a one-to-one mapping from one set of values to another set of values using straightforward rules. For example, if 0/1 corresponds to male/female in one source system, then the gender is stored as male/female in the DW.

A one-to-many transformation is more complex than scalar transformation. As a data element form the source system results in several columns in the DW. Consider the 6 □30 address field (6 lines of 30 characters each), the requirement is to parse it into street address lines 1 and 2, city, state and zip code by applying a parsing algorithm.

The most complex is many-to-many element transformations. Good examples are house holding and individualization. This is achieved by using candidate keys and fuzzy matching to determine which individuals are the same individuals, and which individuals go in the same household and so on. This is a very complex transformation and will be discussed in BSN lecture.

Rigidity and unavailability of legacy systems

- Very difficult to add logic to or increase performance of legacy systems.
- Utilization of expensive legacy systems is optimized.
- Therefore, want to off-load transformation cycles to open systems environment.
- This often requires new skill sets.

Legacy systems typically mean mainframe type systems that are very expensive to buy and operate. We are talking about CPU cycles as well as storage, but mainly computing CPU cycles. The mainframe machines are usually 110% utilized in most businesses. The reason being, businesses want the “most bang for the buck”, so they keep on putting more and more load on the machines.

Consider an expensive ERP system, when is the system available for the processing of DWH

transformations? Probably never, as the system utilization has already been maxed up. So the big question is where to do the transformations? Where do the CPU cycles come from? And when the legacy data is in a mainframe environment, typically the cost of transformation is very high because of scarcity and cost of legacy cycles. But you don't want the cost of constructing the DWH to exceed the value of the DWH. So you need to think of ways to move the cycles to more cost efficient environments.

Therefore, in case of legacy environment, generally you want to off-load the transformation cycles to open systems environment. Open system i.e. Intel chipset, operating systems like Unix and NT etc. as it results in a much more cost effective approach than doing things in a mainframe type of environment. However, this may sometimes be problematic for some organizations requiring certain new skill sets. For example the IT department of a bank may have only used a mainframe for day-to-day operations, and moving on to open systems may be painful for them due to the learning curve etc.

It is NOT simply moving the data to open systems; there are lots of issues due to legacy systems, simple as well as complex. For example, in the mainframe environment you have EBCDIC character representation and in open systems it is always ASCII, so you have to do the conversion. In mainframe environment there is packed decimal encoding, zone decimal and weird date formats that are not understandable in the open systems environment. So you have to have utilities for doing the conversion and so on before the required transformation takes place.

Volume of legacy data

- Talking about not weekly data, but data spread over years.
- Historical data on tapes that are serial and very slow to mount etc.
- Need lots of processing and I/O to effectively handle large data volumes.
- Need efficient interconnect bandwidth to transfer large amounts of data from legacy sources to DWH.

The volume of legacy data is typically very large. It is not just going back and getting tape for one month of billing data, but getting billing data for as much as one could get hold of (say), three years of billing data. Imagine the volume of work, lots of processing and I/Os especially

when the data is on tapes. This would involve tape mounts, and manual tape mounts is painstakingly slow. It takes forever to mount hundreds and hundreds of tapes. The IT people will hate you after you are through with tape mounts and data retrieval.

Assuming transformation takes place on the mainframe; the next issue is to move large volumes of data from the mainframe into the DWH requiring network bandwidth. The data may have to be moved across the LAN, and maybe even across the WAN which is more painful. Therefore, the DWH architects need to pay a lot of attention to capacity planning issues of the DWH i.e. how to size the DWH, how much is it going to cost to extract all the data, prepare the data, and move the data from a network capacity planning perspective. Ensure that from a capacity planning point of view all these aspects are considered.

Web scrapping

- Lot of data in a web page, but is mixed with a lot of "junk".
- Problems:
    - Limited query interfaces
- Fill in forms
    - "Free text" fields
- E.g. addresses
    - Inconsistent output
- i.e., html tags which mark interesting fields might be different on
- different pages.
    - Rapid change without notice.

During the last decade the web has grown from a curiosity to a required information repository. From news headlines and stories, financial data and reports, entertainment events, online shopping and business to business commerce, the Web has it all. The promise of the web is that all of this information will be easily maintained, monitored, queried, and browsed. Some of the promise has already been realized but much remains to be done. Web browsers (such as Internet Explorer and Netscape) have been very successful at displaying web-based information on a computer monitor.

However, there remains a divide between content presented on web pages and the ability to extract that content and make it actionable. What companies and individuals need is a method to extract or mine only that data or content in which they are interested. Furthermore, they need to be able to take that mined content, apply different kinds of data processing, and make it available for use in different applications, and have it delivered to a database, email, spreadsheet, mobile phone, or whatever delivery destination device is available.

Web scrapping is a process of applying screen scrapping techniques to the web. There are several web scrapping products in the market and target business users who want to creatively use the data, not write complex scripts. Some of the uses of scrapping are:

- Building contact lists
- Extracting product catalogs
- Aggregating real-estate info
- Automating search Ad listings
- Clipping news articles etc.

Beware of data quality (or lack of it)

- Data quality is always worse than expected.
- Will have a couple of lectures on data quality and its management.
- It is not a matter of few hundred rows.
- Data recorded for running operations is not usually good enough for decision support.
- Correct totals don’t guarantee data quality.
- Not knowing gender does not hurt POS.
- Centurion customers popping up.

Personal word of warning, the data quality will be always be worse than you expect, you can count on this. Everybody says that other people have dirty data (like everyone thinks they will win the lottery and other people will have an accident), but my data is clean. I don’t believe you; your data is always dirty. You need to allocate time and resources to facilitate the data cleanup. In the next series of lectures, we will talk about a methodology using TQM techniques and apply to DWH data quality. There is a whole methodology on how to deliver data quality in a data

warehouse environment. This is just a warning, data quality is always worse than you think it is. You have used data for transaction processing; it does not mean it is good for decision support.

Everybody always underestimate how dirty their data is. It is not a few hundred rows of dirty data. Data is always dirtier than you think. I have not seen a single business environment that has data that is as clean as it should be. Especially those things required for decision making. Most people concentrate on data quality on numbers, making sure the amounts are right, make sure the accounts are right those kinds of things because you need those to print statements to do business. But for decision making I need different things. I need to know the gender of the customer to understand my market place. I don't need to know the gender of the customer to process a transaction. So basically nobody cares about it, and they put any garbage they want in there and that becomes a problem later on. Absolutely do not assume the data is clean, assume the opposite, assume the data is dirty and help prove it otherwise.

ETL vs. ELT

There are two fundamental approaches to data acquisition:

ETL: Extract, Transform, Load in which data transformation takes place on a separate transformation server.

ELT: Extract, Load, Transform in which data transformation takes place on the data warehouse server.

Combination of both is also possible

You want to have a data flow driven architecture that you understand these points i.e. what data is required and how does it flow in the system driven by the meta data. Once we have those data flows then we can parallelize. You leverage pre-packaged tool for the transformation steps whenever possible. But then again have to look at the practical realities of the market place. That's why I say whenever possible.Whenever possible partially means whenever economically feasible.

In the architecture, ETL presents itself very suitable for data parallelism. Because I have got tons

of data and I want to apply operations consistently across all of that data. So data parallelism is almost always used and you may use pipeline parallelism assuming I do not need too many sources so on. The difficulty of doing this parallelization by hand is very high. Because if you want to write a parallel program say a parallel C program, like parallel pro or something like that, it is very difficult. Again it is not just writing a parallel program, but you also need to make sure it works with check-points restarts and/or error conditions and all those things. I suspect that in your environment, but this is true for NADRA and PTML, that use the database to do the parallelization. So a different kind of approach, what if you are not willing to buy a tool that costs US$ 200,000?

This is a different kind of approach called ELT which is Extract Load Transform. We extract, we load into the database and then we transform in the parallel database. Then we get all the parallelism for free, because you already have a parallel database. You don't have to buy a separate tool in order to get the parallelization.

ETL Detail: Data Extraction & Transformation

Learning Goals

- Extracting of Data
- CDC in Modern Systems
- CDC in Legacy Systems
- CDC Advantages
- Major Transformation Types

Oftentimes, data warehousing involves the extraction and transportation of relational data from one or more source databases, into the data warehouse for analysis. Change Data Capture (CDC) quickly identifies and processes only the data that has changed, not entire tables, and makes the change data available for further use.

Without Change Data Capture, database extraction is a cumbersome process in which you move the entire contents of tables into flat files, and then load the files into the data warehouse. This ad hoc approach is expensive in a number of ways.

Extracting Changed Data

Incremental data extraction

Incremental data extraction i.e. what has changed, say during last 24 hrs if considering nightly extraction.

Efficient when changes can be identified

This is efficient, when the small changed data can be identified efficiently.

Identification could be costly

Unfortunately, for many source systems, identifying the recently modified data may be difficult or effect operation of the source system.

Very challenging

Change Data Capture is therefore, typically the most challenging technical issue in data extraction.

Two CDC sources

? Modern systems

? Legacy systems

If a data warehouse extracts data from an operational system on a nightly basis, then the data warehouse requires only the data that has changed since the last extraction (that is, the data that has been modified in the past 24 hours).

The changed data resulting from INSERT, UPDATE, and DELETE operations made to user tables is recorded. For some environments, the change data is then stored in a database object called a *change table*, and the changed data is made available to applications in a controlled way. When it is possible to efficiently identify and extract only the most recently changed data, the extraction process (as well as all downstream operations in the ETL process) can be much more efficient, because it must extract a much smaller volume of data.

Unfortunately, for many source systems, identifying the recently modified data may be difficult or intrusive to the operation of the system. Change Data Capture is typically the most challenging

technical issue in data extraction.

CDC in Modern Systems

? Time Stamps
? Works if timestamp column present
? If column not present, add column
? May not be possible to modify table, so add triggers

? Triggers
? Create trigger for each source table
? Following each DML operation trigger performs updates
? Record DML operations in a log

? Partitioning
? Table range partitioned, say along date key
? Easy to identify new data, say last week's data

Several techniques are used for implementing changed data capture in currently used state of the art source systems:
? Timestamps
? Partitioning
? Triggers

These techniques are based upon the characteristics of the source systems, or may require modifications to the source systems. Thus, each of these techniques must be carefully evaluated by the owners of the source system prior to implementation.

Timestamps

The tables in some operational systems have timestamp columns. The timestamp specifies the time and date that a given row was last modified. If the tables in an operational system have columns containing timestamps, then the latest data can easily be identified using the timestamp columns. If the timestamp information is not available in an operational source system, you will not always be able to modify the system to include timestamps. Such modification would require,

first, modifying the operational system's tables to include a new timestamp column and then creating a trigger to update the timestamp column following every operation that modifies a given row.

Triggers

Triggers can be created in operational systems to keep track of recently updated records. They can then be used in conjunction with timestamp columns to identify the exact time and date when a given row was last modified. You do this by creating a trigger on each source table that requires change data capture. Following each DML statement that is executed on the source table, this trigger updates the timestamp column with the current time. Thus, the timestamp column provides the exact time and date when a given row was last modified. Then, whenever any modifications are made to the source table, a record is inserted into the materialized view log indicating which rows were modified

Partitioning

Some source systems might use range partitioning, such that the source tables are partitioned along a date key, which allows for easy identification of new data. For example, if you are extracting from an orders table, and the orders table is partitioned by week, then it is easy to identify the current week's data.

CDC in Legacy Systems

- Changes recorded in tapes Changes occurred in legacy transaction processing are recorded on the log or journal tapes.

- Changes read and removed from tapes Log or journal tape is read and the update/transaction changes are stripped off for movement into the data warehouse.
- Problems with reading a log/journal tape are many:
    - Contains lot of extraneous data
    - Format is often arcane
    - Often contains addresses instead of data values and keys

- Sequencing of data in the log tape often has deep and complex implications
- Log tape varies widely from one DBMS to another.

The log or journal tape is used for backup and recovery purposes and is a normal part of the legacy DBMS infrastructure. CDC uses the legacy log tape as a means of capturing and identifying the update/ transaction changes that have occurred throughout the online day. CDC requires that the log or journal tape be read and the update/transaction changes be stripped out for the purpose of preparing the data for movement into the data warehouse. But reading a log/journal tape is no small matter. There are many obstacles that are in the way:

- The log tape contains lot of extraneous data,
- The log tape format is often arcane,
- The log tape often contains addresses instead of data values and keys,
- The sequencing of data in the log tape often has deep and complex implications,
- The log tape reflects the idiosyncrasies of the DBMS and varies widely from one DBMS to another, etc.

CDC Advantages

1. No incremental on-line I/O required for log tape
2. The log tape captures all update processing
3. Log tape processing can be taken off-line.
4. No haste to make waste.
5. No loss of history
6. Flat files NOT required
7. Immediate.

Some of the advantages of CDC are;

1. Database extraction from INSERT, UPDATE, and DELETE operations occurs immediately, at the same time the changes occur to the source tables.
2. In the absence of CDC database extraction is marginal at best for INSERT operations,

And problematic for UPDATE and DELETE operations, because the data is no longer in the

table.

3. Stages data directly to relational tables; there is no need to use flat files. In the absence of CDC, the entire contents of tables are moved into flat files.
4. Log tape processing requires no extra I/O in the middle of online processing, as does replication processing. (Strictly speaking, log tape processing does require I/O. But The I/O was going to be spent whether the log tape was used for data warehouse refreshment or not. Therefore the marginal amount of extra I/O is nil when it comes to using the log tape for data warehouse refreshment.)
5. Another advantage of CDC is that the log tape captures all update processing. There is no need to go back and redefine parameters when there is a change made either to the data warehouse or to the legacy systems environment. The log tape is as basic and as stable as you can get.
6. The sixth and perhaps the most important reason for using the log tape is that the log tape processing can be taken off line. There is no need for any demands to be made against the native legacy DBMS when the log tape is used.
7. Finally data is able to be integrated and transformed "in-flight". Once the update/transaction data has been pulled from the log tape, the DWH is free to re-sequence, reformat, convert, merge, summarize, etc. There is no unnecessary haste or rush to get the data into the data warehouse with log tape processing. Therefore there is the opportunity to properly integrate the update/ transaction data before passing the data into the data warehouse.
8. Stated differently, unlike the data warehouse that results from raw transactions being stacked into a data warehouse from replication, with CDC the update/transaction data is able to be properly integrated before passing into the data warehouse.

Major Transformation Types

1. Format revision
2. Decoding of fields
3. Calculated and derived values

4. Splitting of single fields
5. Merging of information
6. Character set conversion
7. Unit of measurement conversion
8. Date/Time conversion
9. Summarization
10. Key restructuring
11. Duplication

Irrespective of the variety and complexity of the source operational systems, and regardless of the extent of your data warehouse, you will find that most of your data transformation functions break down into a few basic tasks. We have already discussed them. When we consider a particular set of extracted data structures, you will find that the transformation functions you need to perform on this set may done by doing a combination of the basic tasks discussed. Now let us consider specific types of transformation functions. These are the most common transformation types:

Major Transformation Types

1. Format revision
2. Decoding of fields

Format Revisions: You will come across these quite often. These revisions include changes to the data types and lengths of individual fields 12 digit new NID. In your source systems, product package types may be indicated by codes and names in which the fields are numeric and text data types. Again, the lengths of the package types may vary among the different source systems. It is wise to standardize and change the data type to text to provide values meaningful to the users.

Decoding of Fields: This is also a common type of data transformation. When you deal with multiple source systems, you are bound to have the same data items described by a plethora of field values. The classic example is the coding for gender, with one source system using 1 and 2 for male and female and another system using M and F. Also, many legacy systems are notorious for using cryptic codes to represent business values. What do the codes AC, IN, RE, and SU mean in a customer file? You need to decode all such cryptic codes and change these into values

that make sense to the users. Change the codes to Active, Inactive, Regular, and Suspended.

Calculated and Derived Values: What if you want to keep profit margin along with sales and cost amounts in your data warehouse tables? (Briefly touched in de- normalization). The extracted data from the sales system contains sales amounts, sales units, and operating cost estimates by product. You will have to calculate the total cost and the profit margin before data can be stored in the data warehouse. Age and GPA are examples of derived fields. **Splitting of Single Fields:** Earlier legacy systems stored names and addresses of customers and employees in large text fields. The first name, middle name, and family name were stored as a large text in a single field. Similarly, some earlier systems stored road/street, sector, and city data together in a single field. You need to store individual components of names and addresses in separate fields in your data warehouse for two reasons. First, you may improve the operating performance by indexing on individual components. Second, your users may need to perform analysis by using individual components such as road, sector, city etc.

Major Transformation Types

- Merging of information
  Not really means combining columns to create one column.Info for product coming from different sources merging it into single entity.
- Character set conversion
  For PC architecture converting legacy EBCIDIC to ASCII
- Unit of measurement conversion
  For companies with global branches Km vs. mile or lb vs. Kg
- Date/Time conversion
  November 14, 2005 as 11/14/2005 in US and 14/11/2005 in the British format. This date may be standardized to be written as 14 NOV 2005.

Merging of Information: This is not quite the opposite of splitting of single fields. This type of data transformation does not literally mean the merging of several fields to create a single field of data. For example, information about a product may come from different data sources. The product Code and description may come from one data source. The relevant package types may be found in another data source. The cost data may be from yet another source. In this case,

merging of information denotes the combination of the product code, description, package types, and cost into a single entity.

Character Set Conversion: This type of data transformation relates to the conversion of character sets to an agreed standard character set for textual data in the data warehouse. If you have mainframe legacy systems as source systems, the source data from these systems will be in EBCDIC characters. If PC-based architecture is the choice for your data warehouse, then you must convert the mainframe EBCDIC format to the ASCII format. When your source data is on other types of hardware and operating systems, you are faced with similar character set conversions.

Conversion of Units of Measurements: Many companies today have global branches.
Measurements in many European countries are in metric units. If your company has overseas operations, you may have to convert the metrics so that the numbers may all be in one standard unit of measurement.

Date/Time Conversion: This type relates to representation of date and time in standard formats. For example, the American and the British date formats may be standardized to an international format. The date of November 14, 2005 is written as 11/14/2005 in the
U.S. format and as 14/11/2005 in the British format. This date may be standardized to be written as 14 NOV 2005.

Major Transformation Types

Aggregation & Summarization
Adding like values

How they are different?
Summarization with calculation across business dimension is aggregation. Example Monthly compensation = monthly sale + bonus

Why both are required?
Grain mismatch (don't require, don't have space)
Data Marts requiring low detail

Detail losing its utility

Data stored in data warehouses is usually summarized and aggregated at some level because of the vast size of most data warehouses coupled with the analytical processing that occurs on the warehouse data. Although summarization and aggregation are sometimes used interchangeably, you will find a subtle difference between the two.

Summarization is the addition of like values along one or more business dimensions. An example of summarization is adding up detail revenue values by day to arrive at weekly totals (or by week to arrive at monthly totals, by month to arrive at quarterly totals, and so on). **Aggregation** refers to a summarization coupled with a calculation across different business elements. An example of aggregation is the addition of bimonthly salary to monthly commission and bonus to arrive at monthly employee compensation values.

Depending on the data requirements of the warehouse, both summarization and aggregation can be deployed during data transformation. Summarization and aggregation are typically used for the following reasons:

? They are required when the lowest level of detail stored in the data warehouse is at a higher level than the detail arriving from the source. This situation occurs when data warehouse queries do not require the lowest level of detail or sometimes when sufficient disk space is not available to store all the data for the time frame required by the data warehouse.

? They can be used to populate data marts from the data warehouse where the data mart does not require the same level of detail as is stored in the warehouse.

? They can be used to roll up detail values when the detail is removed from the warehouse because it is not being used or because it has aged past its useful life in the data warehouse

Major Transformation Types

Key Restructuring: While extracting data from your input sources, look at the primary keys of the extracted records. You will have to come up with keys for the fact and dimension tables based on the keys in the extracted records. In the example shown in the Table 18.1, the product code in this organization is structured to have inherent meaning. If you use this product code as the primary key, there would be problems. If the product is moved to another warehouse, the

warehouse part of the product key will have to be changed. This is a typical problem with legacy systems. When choosing keys for your data warehouse database tables, avoid such keys with built-in meanings. Transform such keys into generic keys generated by the system itself. This is called key restructuring.

Reduplications: In a normal client database some clients may be represented by several records for various reasons (i) incorrect or missing data values because of data entry errors (ii) inconsistent value naming conventions because of different entry formats and use of abbreviations such as ONE vs. 1 (iii) incomplete information because data is not captured or available (iv) clients do not notify change of address and (v) clients misspell their names or give false address or incorrect information about themselves. As a result we encounter situations where several records may refer to the same real world entity while not being syntactically equivalent.

In your data warehouse, you want to keep a single record for one customer and link all the duplicates in the source systems to this single record. This process is called deduplication of the customer file. Employee files and, sometimes, product master files have this kind of duplication problem.

Data content defects

Domain value redundancy
Non-standard data formats
Non-atomic data values
Multipurpose data fields
Embedded meanings
Inconsistent data values
Data quality contamination

These are just some of the data content defects, there are and can be many more. We will discuss them in detail in the next lecture.