Chapter 4
Data Warehouse technologies and implementations

Incomplete, noisy, and inconsistent data are commonplace properties of large real-world databases and data warehouses. Incomplete data can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Other data may not be included simply because it was not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions. Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the history or modifications to the data may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.

ETL (Extract-Transform-Load)

ETL comes from Data Warehousing and stands for Extract-Transform-Load. ETL covers a process of how the data are loaded from the source system to the data warehouse. Currently, the ETL encompasses a cleaning step as a separate step. The sequence is then Extract-Clean-Transform-Load. In the ETL process, three separate functions are combined into one development tool:

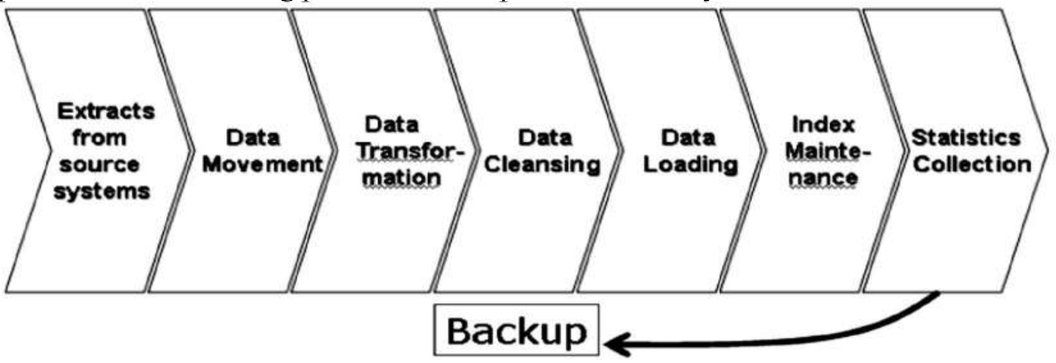
1. **Extract** - Reads data from a specified source and extracts a desired subset of data.
2. **Transform** - Uses rules or lookup tables, or creating combinations with other data, to convert source data to the desired state.
3. **Load** - Writes the resulting data to a target database

Furthermore, these tools and utilities include the following functions:

- **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources
- **Data cleaning**, which detects errors in the data and rectifies them when possible
- **Data transformation**, which converts data from legacy or host format to warehouse format
- **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions
- **Refresh**, which propagates the updates from the data sources to the warehouse

Overview

ETL, Short for *extract, transform, and load a*re the database functions that are combined into one tool. ETL is used to migrate data from one database to another, to form data marts and data warehouses and also to convert databases from one format or type to another. To get data out of the source and load it into the data warehouse –ETL is simply a process of copying data from one database to other. Data is extracted from an OLTP database, transformed to match the data warehouse schema and loaded into the data warehouse database. Many data warehouses also incorporate data from non-OLTP systems such as text files, legacy systems, and spreadsheets; such data also requires extraction, transformation, and loading. ETL is independent yet interrelated steps. It is important to look at the big picture. Data acquisition time may include…



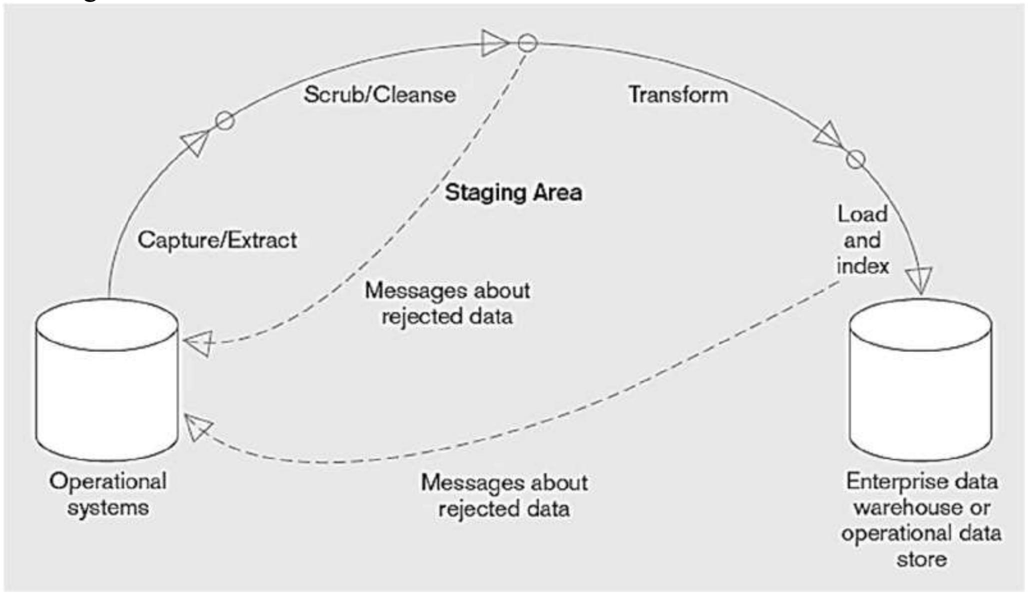
ETL is often a complex combination of process and technology that consumes a significant portion of the data warehouse development efforts and requires the skills of business analysts, database designers, and application developers. It is not a one-time event as new data is added to the Data

Warehouse periodically – i.e. monthly, daily, hourly. Because ETL is an integral, ongoing, and recurring part of a data warehouse. It may be:

- Automated
- Well documented
- Easily changeable

When defining ETL for a data warehouse, it is important to think of ETL as a process, not a physical implementation.

Extraction, Transformation, and Loading (ETL) Processes
Data Extraction
Data Cleansing
Data Transformation
Data Loading
Data Refreshing


Scrub/Cleanse
Transform
Staging Area
Capture/Extract
Load and index
Messages about rejected data
Operational systems
Messages about rejected data
Enterprise data warehouse or operational data store

Data Extraction
Extraction is the operation of extracting data from a source system for further use in a data warehouse environment. This is the first step of the ETL process. After the extraction, this data can be transformed, cleansed and loaded into the data warehouse. Data is extracted from heterogeneous data sources. Each data source has its distinct set of characteristics that need to be managed and integrated into the ETL system in order to effectively extract data. ETL process needs to effectively integrate systems that have different:

- DBMS
- Operating Systems
- Hardware
- Communication protocols
- Need to have a logical data map before the physical data can be transformed

The logical data map describes the relationship between the extreme starting points and the extreme ending points of your ETL system usually presented in a table or spreadsheet. The content of the logical data mapping document has been proven to be the critical element required to efficiently plan ETL processes. The table type gives us our queue for the ordinal position of our data load processes—first dimensions, then facts. This table must depict, without question, the course of action involved in the transformation process. The transformation can contain anything from the absolute solution to nothing at all. Most often, the transformation can be expressed in SQL. The SQL may or may not be the complete statement.

Some ETL Tools: **Oracle Warehouse Builder (OWB), IBM Information Server (Ascential), SAS Data Integration Studio, Oracle Data Integrator (Sunopsis), DB2 Warehouse Edition, etc.**

Data Cleansing

There are many possible reasons for noisy data (having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Errors in data transmission can also occur. There may be technology limitations, such as limited buffer size for coordinating synchronized data transfer and consumption. Incorrect data may also result from inconsistencies in naming conventions or data codes used or inconsistent formats for input fields, such as *date*. Duplicate tuples also require data cleaning.

Data cleaning(or data cleansing) routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies. If users believe the data are dirty, they are unlikely to trust the results of any data mining that has been applied to it. Furthermore, dirty data can cause confusion for the mining procedure, resulting in unreliable output. Although most mining routines have some procedures for dealing with incomplete or noisy data, they are not always robust. Instead, they may concentrate on avoiding over fitting the data to the function being modeled. Therefore, a useful preprocessing step is to run your data through some data cleaning routines.

Data warehouse contains data that is analyzed for business decisions. Source systems contain "dirty data" that must be cleansed. More data and multiple sources could mean more errors in the data and harder to trace such errors which results in incorrect analysis. So there are enormous problem, as most data is dirty. Data Warehouse is NOT just about arranging data, but should be clean for overall health of organization. This process is sometime called as **Data Scrubbing** or **Cleaning**. ETL software contains rudimentary data cleansing capabilities. Specialized data cleansing software is often used. Leading data cleansing vendors include Vality (Integrity), Harte-Hanks (Trillium), and Firstlogic (i.d.Centric).

Reasons

- Dummy Values
- Absence of Data
- Multipurpose Fields
- Cryptic Data
- Contradicting Data
- Inappropriate Use of Address Lines
- Violation of Business Rules
- Reused Primary Keys,
- Non-Unique Identifiers
- Data Integration Problems

The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

- Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
- Convert null values into standardized Not Available/Not Provided value
- Convert phone numbers, ZIP codes to a standardized form
- Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
- Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

Data Transformation

The next operation in the ETL process is the Data Transformation. It is the main step where the ETL adds value. It actually changes data and provides guidance whether data can be used for its intended purposes and performed in staging area. It applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules. The major tasks performed during this phase vary depending on the application; however, the basic tasks are:

1. Selection
2. Splitting/Joining
3. Conversion
4. Summarization
5. Enrichment

Data Loading

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency. Most loads involve only change data rather than a bulk reloading of all of the data in the warehouse. Data are physically moved to the data warehouse. The loading takes place within a "load window". The trend is to near real-time updates of the data warehouse as the warehouse is increasingly used for operational applications. The loading process can be broken down into 2 different types:

- Initial Load: Consists of populating tables in warehouse schema and verifying data readiness
- Continuous Load (loading over time): Must be scheduled and processed in a specific order to maintain integrity, completeness, and a satisfactory level of trust.

Data Refreshing

During the refresh step, it is necessary to propagate updates from sources to the warehouse and administrator set refreshing depending on user needs and traffic. Data refreshing is done:

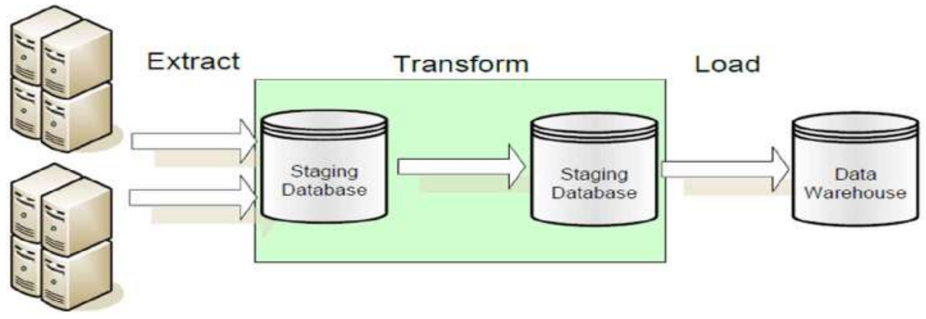
- periodically (e.g., every night, every week) or after significant events
- on every update: not warranted unless warehouse data require current data (up to the minute stock quotes)
- refresh policy set by administrator based on user needs and traffic
- possibly different policies for different sources

Issues:

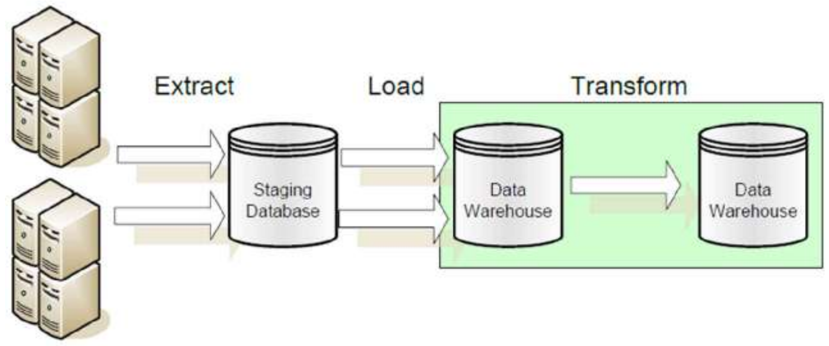
- ❖ when to refresh
- ❖ how to refresh -- refresh techniques

ETL vs. ELT

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.



Process

Extract

The Extract step covers the data extraction from the source system and makes it accessible for further processing. The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible. The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

- Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.
- Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.
- Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.

When using Incremental or Full extracts, the extract frequency is extremely important. Particularly for full extracts; the data volumes can be in tens of gigabytes.

Clean

The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

- Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
- Convert null values into standardized Not Available/Not Provided value
- Convert phone numbers, ZIP codes to a standardized form
- Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
- Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

Transform

The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Managing ETL Process
The ETL process seems quite straight forward. As with every application, there is a possibility that the ETL process fails. This can be caused by missing extracts from one of the systems, missing values in one of the reference tables, or simply a connection or power outage. Therefore, it is necessary to design the ETL process keeping fail-recovery in mind.

Staging
It should be possible to restart, at least, some of the phases independently from the others. For example, if the transformation step fails, it should not be necessary to restart the Extract step. We can ensure this by implementing proper staging. Staging means that the data is simply dumped to the location (called the Staging Area) so that it can then be read by the next processing phase. The staging area is also used during ETL process to store intermediate results of processing. This is ok for the ETL process which uses for this purpose. However, the staging area should be accessed by the load ETL process only. It should never be available to anyone else; particularly not to end users as it is not intended for data presentation to the end-user. May contain incomplete or in-the-middle-of-the-processing data.

Extraction of Data
During extraction, the desired data is identified and extracted from many different sources, including database systems and applications. Very often, it is not possible to identify the specific subset of interest; therefore more data than necessary has to be extracted, so the identification of the relevant data will be done at a later point in time. Depending on the source system's capabilities (for example, operating system resources), some transformations may take place during this extraction process. The size of the extracted data varies from hundreds of kilobytes up to gigabytes, depending on the source system and the business situation. The same is true for the time delta between two (logically) identical extractions: the time span may vary between days/hours and minutes to near real-time. Web server log files, for example, can easily grow to hundreds of megabytes in a very short period of time.

Transportation of Data
After data is extracted, it has to be physically transported to the target system or to an intermediate system for further processing. Depending on the chosen way of transportation, some transformations can be done during this process, too. For example, a SQL statement which directly accesses a remote target through a gateway can concatenate two columns as part of the SELECT statement.
The emphasis in many of the examples in this section is scalability. Many long-time users of Oracle Database are experts in programming complex data transformation logic using PL/SQL. These chapters suggest alternatives for many such data manipulation operations, with a particular emphasis on implementations that take advantage of Oracle's new SQL functionality, especially for ETL and the parallel query infrastructure.

ETL Tool Implementation
When you are about to use an ETL tool, there is a fundamental decision to be made: will the company build its own data transformation tool or will it use an existing tool? Building your own data transformation tool (usually a set of shell scripts) is the preferred approach for a small number of data sources which reside in storage of the same type. The reason for that is the effort to implement the necessary transformation is little due to similar data structure and common system architecture. Also, this approach saves licensing cost and there is no need to train the staff in a new tool. This approach, however, is dangerous from the TOC point of view. If the transformations become more sophisticated during the time or there is a need to integrate other systems, the

complexity of such an ETL system grows but the manageability drops significantly. Similarly, the implementation of your own tool often resembles re-inventing the wheel.
There are many ready-to-use ETL tools on the market. The main benefit of using off-the-shelf ETL tools is the fact that they are optimized for the ETL process by providing connectors to common data sources like databases, flat files, mainframe systems, xml, etc. They provide a means to implement data transformations easily and consistently across various data sources. This includes filtering, reformatting, sorting, joining, merging, aggregation and other operations ready to use. The tools also support transformation scheduling, version control, monitoring and unified metadata management. Some of the ETL tools are even integrated with BI tools.

Some of the Well Known ETL Tools
The most well-known commercial tools are Ab Initio, IBM Info Sphere Data Stage, Informatica, Oracle Data Integrator and SAP Data Integrator. There are several open source ETL tools, among others Apatar, CloverETL, Pentaho and Talend.

Data Integration
Data integration involves combining data from several disparate sources, which are stored using various technologies and provide a unified view of the data. Data integration becomes increasingly important in cases of merging systems of two companies or consolidating applications within one company to provide a unified view of the company's data assets. The later initiative is often called a data warehouse.
Probably the most well-known implementation of data integration is building an enterprise's data warehouse. The benefit of a data warehouse enables a business to perform analyses based on the data in the data warehouse. This would not be possible to do on the data available only in the source system. The reason is that the source systems may not contain corresponding data, even though the data are identically named, they may refer to different entities.

Data Integration Areas
Data integration is a term covering several distinct sub-areas such as:

- Data warehousing
- Data migration
- Enterprise application/information integration
- Master data management

This article concentrates on the process of data integration. More detailed information about the above areas can be found in related articles.

Challenges of Data Integration
At first glance, the biggest challenge is the technical implementation of integrating data from disparate often incompatible sources. However, a much bigger challenge lies in the entirety of data integration. It has to include the following phases:

Design

- The data integration initiative within a company must be an initiative of business, not IT. There should be a champion who understands the data assets of the enterprise and will be able to lead the discussion about the long-term data integration initiative in order to make it consistent, successful and beneficial.
- Analysis of the requirements (BRS), i.e. why is the data integration being done, what are the objectives and deliverables. From what systems will the data be sourced? Is all the data available to fulfill the requirements? What are the business rules? What is the support model and SLA?
- Analysis of the source systems, i.e. what are the options of extracting the data from the systems (update notification, incremental extracts, full extracts), what is the required/available frequency of the extracts? What is the quality of the data? Are the required

data fields populated properly and consistently? Is the documentation available? What are the data volumes being processed? Who is the system owner?
- Any other non-functional requirements such as data processing window, system response time, estimated number of (concurrent) users, data security policy, backup policy.
- What is the support model for the new system? What are the SLA requirements?
- And last but not least, who will be the owner of the system and what is the funding of the maintenance and upgrade expenses?
- The results of the above steps need to be documented in form of SRS document, confirmed and signed-off by all parties which will be participating in the data integration project.

Implementation
Based on the BRS and SRS, a feasibility study should be performed to select the tools to implement the data integration system. Small companies and enterprises which are starting with data warehousing are faced with making a decision about the set of tools they will need to implement the solution. The larger enterprise or the enterprises which already have started other projects of data integration are in an easier position as they already have experience and can extend the existing system and exploit the existing knowledge to implement the system more effectively. There are cases, however, when using a new, better suited platform or technology makes a system more effective compared to staying with existing company standards. For example, finding a more suitable tool which provides better scaling for future growth/expansion, a solution that lowers the implementation/support cost, lowering the license costs, migrating the system to a new/modern platform, etc.
Testing
Along with the implementation, the proper testing is a must to ensure that the unified data are correct, complete and up-to-date.
Both technical IT and business needs to participate in the testing to ensure that the results are as expected/required. Therefore, the testing should incorporate at least Performance Stress test (PST), Technical Acceptance Testing (TAT) a nd User Acceptance Testing (UAT) PST, TAT (Technical Acceptance Testing), UAT (User Acceptance Testing).

Data Integration Techniques
There are several organizational levels on which the integration can be performed. As we go down the level of automated integration increases.
Manual Integration or Common User Interface - users operate with all the relevant information accessing all the source systems or web page interface. No unified view of the data exists.
Application Based Integration - requires the particular applications to implement all the integration efforts. This approach is manageable only in case of very limited number of applications.
Middleware Data Integration - transfers the integration logic from particular applications to a new middleware layer. Although the integration logic is not implemented in the applications anymore, there is still a need for the applications to partially participate in the data integration.
Uniform Data Access or Virtual Integration - leaves data in the source systems and defines a set of views to provide and access the unified view to the customer across whole enterprise. For example, when a user accesses the customer information, the particular details of the customer are transparently acquired from the respective system. The main benefits of the virtual integration are nearly zero latency of the data updates propagation from the source system to the consolidated view, no need for separate store for the consolidated data. However, the drawbacks include limited possibility of data's history and version management, limitation to apply the method only to 'similar' data sources (e.g. same type of database) and the fact that the access to the user data generates extra load on the source systems which may not have been designed to accommodate.
Common Data Storage or Physical Data Integration - usually means creating a new system which keeps a copy of the data from the source systems to store and manage it independently of the original system. The most well know example of this approach is called Data Warehouse (DW). The benefits comprise data version management, combining data from very different sources (mainframes,

databases, flat files, etc.). The physical integration, however, requires a separate system to handle the vast volumes of data.

Data Migration
Data Migration is the process of transferring data from one system to another while changing the storage, database or application. In reference to the ETL (Extract-Transform-Load) process, data migration always requires at least Extract and Load steps.
Typically data migration occurs during an upgrade of existing hardware or transfer to a completely new system. Examples include: migration to or from hardware platform; upgrading a database or migrating to new software; or company-mergers when the parallel systems in the two companies need to be merged into one. There are three main options to accomplish data migration:

1. Merge the systems from the two companies into a brand new one
2. Migrate one of the systems to the other one.
3. Leave the systems as they are but create a common view on top of them - a data warehouse.

Data Migration Challenges
Let us describe the data migration challenges in little more detail. Data migration can be a simple process, however there are challenges one may face in implementation.
Storage Migration
Storage migration can be handled in a manner transparent to the application so long as the application uses only general interfaces to access the data. In most systems this is not an issue. However, careful attention is necessary for old applications running on proprietary systems. In many cases, the source code of the application is not available and the application vendor may not be in market anymore. In such cases storage migration is rather tricky and should be properly tested before releasing the solution into production.

Database Migration
Database migration is rather straight forward, assuming the database is used just as storage. It "only" requires moving the data from one database to another. However, even this may be a difficult task. The main issues one may encounter include:

- Unmatched data types (number, date, sub-records)
- Different character sets (encoding)

Different data types can be handled easily by approximating the closest type from the target database to maintain data integrity. If a source database supports complex data formats (e.g. sub-record), but the target database does not, amending the applications using the database is necessary. Similarly, if the source database supports different encoding in each column for a particular table but the target database does not, the applications using the database need to be thoroughly reviewed.
When a database is used not just as data storage, but also to represent business logic in the form of stored procedures and triggers, close attention must be paid when performing a feasibility study of the migration to target database. Again, if the target database does not support some of the features, changes may need to be implemented by applications or by middleware software.
ETL tools are very well suited for the task of migrating data from one database to another. Using the ETL tools is highly advisable particularly when moving the data between the data stores which do not have any direct connection or interface implemented.

Application Migration
If we take a step back to previous two cases, you may notice that the process is rather straight forward. This however, is extremely uncommon in the case of application migration. The reason is that the applications, even when designed by the same vendor, store data in significantly different formats and structures which make simple data transfer impossible. The full ETL process is a must as the Transformation step is not always straight forward. Of course, application migration can and usually does include storage and database migration as well. The advantage of an ETL tool in this instance is its ready-to-use connectivity to disparate data sources/targets.

Difficulty may occur when migrating data from mainframe systems or applications using proprietary data storage. Mainframe systems use record based formats to store data. Record based formats are easy to handle; however, there are often optimizations included in the mainframe data storage format which complicate data migration. Typical optimizations include binary coded decimal number storage, non-standard storing of positive/negative number values, or storing the mutually exclusive sub-records within a record. Let us consider a library data warehouse as an example. There are two types of publications - books and articles. The publication can be either a book or an article but not both. There are different kinds of information stored for books and articles. The information stored for a book and an article are mutually exclusive. Hence, when storing a book, the data used has a different sub-record format for a book and an article while occupying the same space. Still the data is stored using rather standard encoding. On the Contrary, proprietary data storage makes the Extract step even more complicated. In both cases, the most efficient way to extract data from the source system is performing the extraction in the source system itself; then converting the data into a printable format which can be parsed later using standard tools.

Character Encoding
Most of the systems developed on PC-based platform use ASCII encoding or national extension based on ASCII. The latest one is UTF-8 which keeps ASCII mapping for alpha and numerical characters but enables storage of characters for most of the national alphabets including Chinese, Japanese and Russian. Mainframe systems are mostly based on EBCDIC encoding which is incompatible with ASCII and conversion is required to display the data. ETL tools should support the conversions between character sets, including EBCDIC.

Data Synchronization
Data Synchronization is a process of establishing consistency among systems and subsequent continuous updates to maintain consistency. The word 'continuous' should be stressed here as the data synchronization should not be considered as a one-time task. It is really a process which needs to be planned, owned, managed, scheduled and controlled.

Motivation
Let us present two scenarios in which data synchronization is crucial for an enterprise.

- In any enterprise, there are often at least 10 systems which are sharing the same data - customer data, product data, employee data, customer support systems, billing and invoicing systems, etc. In order to make the company's manufacturing process auditable, each activity needs to be properly logged. For example, if a company is in car manufacturing business, for each car it need to log from which parts the car was assembled, the part's serial numbers, lot numbers, part supplier; the employee id who has mounted the part; and in the end to whom has the car been sold and the car's service history including the service station and possibly even the technicians and spare parts. Each of the company's production systems, however, has a partial piece of information - the pieces that it actually needs for operation. For example, the system logging the car assembly stores the information about employees; similarly it needs to access the information about suppliers and parts in stock. Even though several applications/systems use the same data, the data is captured only by one application. The data then needs to be synchronized to other systems.

- With the advent of the internet and increasing international business, many companies choose to distribute their systems geographically to reduce the latency and cost of the network usage and to increase reliability (by reducing the risk of e.g. natural disaster affecting the location). The systems in all locations, however, do need to have the same data even though the data is modified in several locations in parallel. The data needs to be synchronized across all locations.

Process

Planning

Requirements on the data synchronization should be gathered in the planning phase. This needs to cover the data content, data formats, initial load and frequency of the updates. Non-functional requirements like performance, timing and security should be covered as well.

Ownership

Although the data synchronization idea may come from the IT organization of the company, an owner or champion from the company business is necessary to provide a continuity of the initiative. It is business who will benefit from the data synchronization initiative in the end.

Scheduling

The scheduling and frequency of updates is one of the items which need to be investigated during initial planning phase. Often the requirements change during this time and the schedule updates needs to be revised. Obviously, the granularity of a schedule on which the updates/synchronization is performed cannot be finer than the source system is able to provide. However, the scheduling also needs to take into account performance aspects (see the section Challenges below).

Monitoring

The synchronization process should be monitored to evaluate whether the update schedule and frequency meets the company's needs.
From the technical point of view, the synchronization may be implemented on any level:

- System/Application level
- File level (may include even version control)
- Record level synchronization

Challenges

Data Formats Complexity

As the enterprise grows and evolves, new systems from different vendors are implemented. The data formats for employees, products, suppliers and customers vary among different industries which results not only in building a simple interface between the two applications (source and target), but also in a need to transform the data while passing them to the target application. The data formats, of course, vary from proprietary formats through plain text to xml. Some of the applications provide API to push the data directly. ETL tools can be helpful here.

Real-timeliness

The requirement today is that the systems are real time. Customers want to see what the status of their order in e-shop is; the status of a parcel delivery - a real time parcel tracking; what the current balance on their account is; etc. Enterprises need to have their system real-time updated as well to enable smooth manufacturing process, e.g. ordering material when enterprise is running out stock; synchronizing customer orders with manufacturing process, etc. There are thousands of examples from real life when the real time is becoming either advantage or a must to be successful and competitive.
Main challenge with real time data synchronization is to work with systems which do not provide any API to identify the changes. In such cases, performance may be the limiting factor.

Security

Different systems may have different policies to enforce data security and access levels. Even though the security is maintained correctly in the source system which captures the data, the security and information access privileges must be enforced on the target systems as well to prevent any potential misuse of the information. This is particularly an issue when handling personal information or any piece of confidential information under Non-Disclosure Agreement (NDA). Any intermediate results of the data transfer as well as the data transfer itself must be encrypted.

Data Quality
Maintaining data in one place and sharing with other applications is best practice in managing and improving data quality. This prevents inconsistencies in the data caused by updating the same data in one system.

Performance
The data synchronization process consists basically of five phases:

1. Data extraction from the source/master system
2. Data transfer
3. Data transformation
4. Data transfer
5. Data load to the target system

In case of large data, each of these steps may impact performance. Therefore, the synchronization needs to be carefully planned to avoid any negative impact e.g. during peak processing hours.
Maintenance
As any other process, the synchronization process needs to be monitored to ensure that it is running as scheduled and properly handling any errors during the process of synchronization such as rejected records or malformed data.