Chapter V

Data Warehouse Refreshment

Alkis Simitisis
National Technical University of Athens, Greece

Panos Vassiliadis
University of Ioannina, Greece

Spiros Skiadopoulos
University of Peloponnese, Greece

Timos Sellis
Natiuonal Technical University of Athens, Greece

Abstract

In the early stages of a data warehouse project, the designers/administrators have to come up with a decision concerning the design and deployment of the backstage architecture. The possible options are (a) the usage of a commercial ETL tool or (b) the development of an in-house ETL prototype. Both cases have advantages and disadvantages. However, in both cases the design and modeling of the ETL workflows have the same characteristics. The scope of this chapter is to indicate the main challenges, issues, and problems concerning the manufacturing of ETL workflows, in order to assist the designers/administrators to decide which solution suits their data warehouse project better and to help them construct an efficient, robust, and evolvable ETL workflow that implements the refreshment of their warehouse.

Introduction

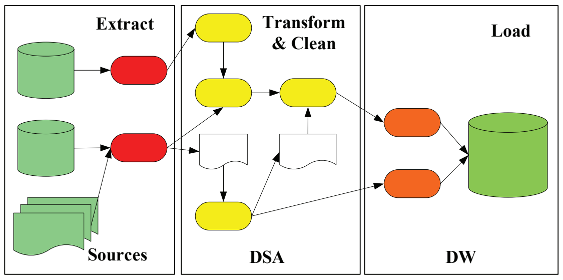
In the past, research has treated data warehouses as collections of materialized views. Although this abstraction is elegant and possibly sufficient for the purpose of examining alternative strategies for view maintenance, it is sufficient enough to describe the structure and contents of a data warehouse in real-world settings. Vassiliadis, Quix, Vassiliou, and Jarke (2001) bring up the issue of *data warehouse operational processes* and deduce the definition of a table in the data warehouse as the outcome of the combination of the processes that populate it. This new kind of definition complements existing approaches, since it provides the operational semantics for the content of a data warehouse table, whereas the existing definitions give an abstraction of its intentional semantics. Indeed, in a typical mediation scheme one would pose a query to a "virtual" data warehouse, dispatch it to the sources, answer parts of it there, and then collect the answers. On the contrary, in the case of data warehouse operational processes, the objective is to carry data from a set of source relations and eventually load them in a target (data warehouse) relation. To achieve this goal, we have to (a) specify data transformations as a workflow and (b) optimize and execute the workflow.

Data warehouse operational processes normally compose a labor intensive workflow and constitute an integral part of the backstage of data warehouse architectures. To deal with this workflow and in order to facilitate and manage the data warehouse operational processes, specialized workflows are used under the general title extraction transformation loading (ETL) workflows. ETL workflows are responsible for the extraction of data from several sources, their cleansing, their customization and transformation, and finally, their loading into a data warehouse.

ETL workflows represent an important part of data warehousing, as they represent the means by which data actually get loaded into the warehouse. To give a general idea of the functionality of these workflows we mention their most prominent tasks, which include:

- The *identification* of relevant information at the source side
- The *extraction* of this information
- The *transportation* of this information to the DSA
- The *transformation* (i.e., customization and integration) of the information coming from multiple sources into a common format
- The *cleaning* of the resulting dataset, on the basis of database and business rules
- The *propagation* and loading of the data to the data warehouse and the refreshment of data marts

Figure 1. The environment of Extraction-Transformation-Loading processes (Simitsis, 2004)



In the sequel, we will adopt the general acronym ETL for all kinds of in-house or commercial tools, and all the aforementioned categories of tasks/processes.

In Figure 1, we abstractly describe the general framework for ETL processes. On the left side, we can observe the original data stores (sources) that are involved in the overall process. Typically, data sources are relational databases and files. The data from these sources are extracted by specialized routines or tools, which provide either complete snapshots or differentials of the data sources. Then, these data are propagated to the data staging area (DSA) where they are transformed and cleaned before being loaded into the data warehouse. Intermediate results, again in the form of (mostly) files or relational tables are part of the data staging area. The data warehouse (DW) is depicted in the right part of Figure 1 and comprises the target data stores, that is, fact tables for the storage of information and dimension tables with the description and the multidimensional rollup hierarchies of the stored facts. The loading of the central warehouse is performed from the loading activities depicted in the right side before the data warehouse data store.

Despite the plethora of commercial solutions that offer ad-hoc capabilities for the creation of an ETL scenario, a designer/administrator needs a concrete method to develop an efficient, robust, and evolvable ETL workflow. Therefore, this chapter intends to point out the main challenges and issues concerning the generic construction of ETL workflows. As an outline, in the rest of the chapter, we proceed with a brief presentation about the state of the art in ETL technology. Afterwards, we discuss why the modeling of ETL workflows is important and we indicate the main problems that arise during all the phases of an ETL process. Moreover, we present a modeling approach for the construction of ETL workflows, which is based on the life cycle of the data warehouse, along with an exemplary research framework named Arktos II. Finally, we list several open research challenges that proclaim ETL as a commodity of future research.

Background

In this section, we present ETL methodologies that are proposed by (a) commercial studies and tools and (b) the research community. Then, we present the reasons and the motives that signify the research on ETL processes is a valid research goal.

State of the Art

- **Commercial studies and tools:** In terms of technological aspects, the main characteristic of the area is the involvement of traditional database vendors with ETL solutions built in the DBMS. The three major database vendors that practically ship ETL solutions "at no extra charge" are pinpointed: Oracle with Oracle Warehouse Builder (Oracle, 2001), Microsoft with Data Transformation Services (Microsoft, 2003), and IBM with the Data Warehouse Center (IBM, 2003). Still, the major vendors in the area are Informatica's Powercenter (Informatica, 2003) and Ascential's DataStage suites (Ascential, 2003) (the latter part of the IBM recommendations for ETL solutions). As a general comment, we emphasize the fact that the former three have the benefit of the minimum cost, because they are shipped with the database, while the latter two have the benefit to aim at complex and deep solutions not envisioned by the generic products. The aforementioned discussion is supported from a second recent study (Gartner, 2003), where the authors note the decline in license revenue for pure ETL tools, mainly due to the crisis of IT spending and the appearance of ETL solutions from traditional database and business intelligence vendors. The Gartner study discusses the role of the three major database vendors (IBM, Microsoft, Oracle) and points out that they slowly start to take a portion of the ETL market through their DBMS-built-in solutions.
- **Research focused specifically on ETL:** The *AJAX* system (Galhardas, Florescu, Shasha & Simon, 2000) is a data cleaning tool developed at INRIA France. It deals with typical data quality problems, such as *the object identity problem* (Cohen, 1999)*, errors due to mistyping,* and *data inconsistencies* between matching records. AJAX provides a framework wherein the logic of a data cleaning program is modeled as a directed graph of data transformations that start from some input source data. AJAX also provides a declarative language for specifying data cleaning programs, which consists of SQL statements enriched with a set of specific primitives to express mapping, matching, clustering, and merging transformations. Finally, a interactive environment is supplied to the user in order to resolve errors and inconsistencies that cannot be automatically handled and support a stepwise refinement design of data cleaning programs. The theoretic foundations of this tool can be found in Galhardas, Florescu, Shasha, and Simon (1999), where apart from the presentation

of a general framework for the data cleaning process, specific optimization techniques tailored for data cleaning applications are discussed.

The *Potter's Wheel* system (Raman & Hellerstein, 2001), is targeted to provide interactive data cleaning to its users. The system offers the possibility of performing several algebraic operations over an underlying dataset. Optimization algorithms are also provided for the CPU usage for certain classes of operators. The general idea behind Potter's Wheel is that users build data transformations in iterative and interactive ways. In the background, Potter's Wheel automatically infers structures for data values in terms of user-defined domains, and accordingly checks for constraint violations. Users gradually build transformations to clean the data by adding or undoing transforms on a spreadsheet-like interface; the effect of a transform is shown at once on records visible on screen. These transforms are specified either through simple graphical operations, or by showing the desired effects on example data values.

- **Data quality and cleaning:** Jarke, List, and Koller (2000) present an extensive review of data quality problems and related literature, along with quality management methodologies. Rundensteiner (1999) offers a discussion on various aspects on data transformations. Sarawagi (2000) presents a similar collection of papers in the field of data cleaning including a survey (Rahm & Hai Do, 2000) that provides an extensive overview of the field, along with research issues and a review of some commercial tools and solutions on specific problems (Borkar, Deshmuk, & Sarawagi, 2000; Monge, 2000). In a related but different context, we would like to mention the IBIS tool (Calì, Calvanese, De Giacomo, Lenzerini, Naggar, & Vernacotola, 2003). IBIS is an integration tool following the global-as-view approach to answer queries in a mediated system. Departing from the traditional data integration literature though, IBIS brings the issue of data quality into the integration process. The system takes advantage of the definition of constraints at the intentional level (e.g., foreign key constraints) and tries to provide answers that resolve semantic conflicts (e.g., the violation of a foreign key constraint).
- **Workflow and process models:** In general, research on workflows is focused around the following reoccurring themes: (a) modeling (Eder & Gruber, 2002; Kiepuszewski, ter Hofstede, & Bussler, 2000; Sadiq & Orlowska, 2000; Van der Aalst, ter Hofstede, Kiepuszewski, & Barros, 2000; Workflow Management Coalition, 1998), where the authors are primarily concerned in providing a metamodel for workflows; (b) correctness issues (Eder & Gruber, 2002; Kiepuszewski et al., 2000; Sadiq & Orlowska, 2000), where criteria are established to determine whether a workflow is well formed, and (c) workflow transformations (Eder & Gruber, 2002; Kiepuszewski et al., 2000; Sadiq & Orlowska, 2000) where the authors are concerned with correctness issues in the evolution of the workflow from a certain plan to another.

- **Applications of ETL workflows in data warehouses:** Finally, the literature reports several efforts (both research and industrial) for the management of processes and workflows that operate on data warehouse systems. Jarke, Quix, Blees, Lehmann, Michalk, and Stierl (1999) describe an industrial effort where the cleaning mechanisms of the data warehouse are employed in order to avoid the population of the sources with problematic data in the first place. The described solution is based on a workflow which employs techniques from the field of view maintenance. Schafer, Becker, and Jarke (2000) describe an industrial effort at Deutche Bank, involving the import/export, transformation and cleaning, and storage of data in a terabyte-size data warehouse. The authors explain also the usage of metadata management techniques, which involves a broad spectrum of applications, from the import of data to the management of dimensional data and more importantly for the querying of the data warehouse. Jarke et al. (2000) present a research effort (and its application in an industrial application) for the integration and central management of the processes that lie around an information system. A metadata management repository is employed to store the different activities of a large workflow, along with important data these processes employ.

Motivation

All engineering disciplines employ blueprints during the design of their engineering artifacts. Modeling in this fashion is not a task with a value, per se; as Booch, Rumbaugh, and Jacobson (1998) mentions "we build models to communicate the desired structure and behavior of our system … to visualize and control the system's architecture … to better understand the system we are building … to manage risk."

Discussing the modeling of ETL workflows is important for several reasons. First, the data extraction, transformation, integration, and loading process is a key part of a data warehouse. The commercial ETL tools that are available on the market the last few years increased their sales from US$101 million dollars in 1998 to US$210 million dollars in 2002, having a steady increase rate of approximately 20.1% each year (Jarke, Lenzerini, Vassiliou, & Vassiliadis, 2003). The same survey indicates that ETL tools are in the third place of the annual sales of the overall components of a data warehouse with the RDBMS sales for data warehouses in the first place (40% each year since 1998) and data marts (25%) in the second place.

Also, ETL processes constitute the major part of a data warehouse environment, resulting in the corresponding development effort and cost. Data warehouse operational processes are *costly* and *critical* for the success of a data warehouse project, and their design and implementation has been characterized as a labor-intensive and lengthy procedure (Demarest, 1999; Shilakes & Tylman, 1998; Vassiliadis, 2000). Several reports mention that most of these processes are constructed through an in-house development procedure that can consume up to 70% of the resources for a data warehouse project (Giga, 2002; Strange, 2002). Complementary reports (Friedman, 2002; Strange, 2002a) address the factors that influence the cost of its implementation and support: (a) *staff* (development, application support teams, on-going support and maintenance, and operations); (b) *computing resources* (dedicated ETL server, disk storage for "temporary" or staging files, CPU use of servers hosting source data, and annual maintenance and support); (c) *tools acquisition* (annual maintenance support and training); (d) *latency* (timeliness of the delivery of data to the target environment impacts the overall effectiveness of BI); and (e) *quality* (flaws in data distilled from ETL processes can severely limit BI adoption). Each of these components directly influences the total cost of ownership of a data warehouse implementation and operation.

For example, Strange (2002) mentions the development of a data warehouse realized in the Fortune 500 financial institution. This development included the support of the data warehouse for applications to perform customer retention analysis, bank loan risk management, customer contact history, and many other applications. There were 100 people on the data warehouse team (approximately 8.5% of the overall IT staff)—55 from ETL, 4 database administrators, 4 architects, 4 systems administrators, 9 BI competency center workers (assisting end users), 5 report writers, 9 managers, and 9 hardware, operating system, and operations support staff members. These 55 individuals were responsible for building and maintaining the ETL process, which includes 46 different source systems. Responsibilities include updates of data marts on a weekly and monthly basis. This does not include staff from operations to support the execution of the ETL processes. They used a large parallel server platform that is consisted of multiple silver nodes (four processors per node) and four terabytes or more of disk storage, at an acquisition cost over three years of US$5 million. The cost of the ETL tool used was US$1 million, excluding the yearly maintenance and support costs.

Moreover, these processes are *important* for the correctness, completeness, and freshness of data warehouse contents, since not only do they facilitate the population of the warehouse with up-to-date data, but they are also responsible for homogenizing their structure and blocking the propagation of erroneous or inconsistent entries.

In addition, these data intensive workflows are quite *complex* in nature, involving dozens of sources, cleaning and transformation activities, and loading facilities. Bouzeghoub, Fabret, and Matulovic (1999) mentions that the data warehouse refreshment process can consist of many different subprocesses, like data cleaning, archiving, transformations, and aggregations, interconnected through a complex schedule. For instance, Adzic and Fiore (2003) report a case study for mobile network traffic data, involving around 30 data flows and 10 sources, while the volume of data rises to about 2 TB, with the main fact table containing about 3 billion records. The throughput of the (traditional) population system is 80 million records per hour for the entire process (compression, FTP of files, decompression, transformation, and loading), on a daily basis, with a loading window of only 4 hours. The request for performance is so pressing that there are processes hard-coded in low level DBMS calls to avoid the extra step of storing data to a target file to be loaded to the data warehouse through the DBMS loader. In general, Strange (2002a) notes that the complexity of the ETL process, as well as the staffing required to implement it, depends mainly on the following variables: (a) the number and variety of data sources; (b) the complexity of transformation; (c) the complexity of integration; and (d) the availability of skill sets. Also, the same report suggests considering "one person per source" as a guide to accomplishing the ETL implementation effectively.

Based on the previous discussion, we can identify key factors underlying the main problems of ETL workflows:

- Vastness of the data volumes
- Quality problems, since data are not always clean and have to be cleansed
- Performance, since the whole process has to take place within a specific time window and it is necessary to optimize its execution time
- Evolution of the sources and the data warehouse can eventually lead to daily maintenance operations

Visualizing and understanding this kind of system is another issue. In fact, traditional modeling approaches need to be reconsidered: we need interactive, multiview modeling frameworks that abstract the complexity of the system and provide complementary views of the system's structure to the designer (apart from simply providing the big picture, like the traditional ER/DFD approaches did). Moreover, we need to be able to manage risk through our modeling artifacts. For example, we would like to answer questions like:

- Which attributes/tables are involved in the population of a certain attribute?
- What part of the scenario is affected if we delete an attribute?
- How good is the design of my ETL workflow?
- Is variant A better than variant B?

Main Thrust of the Chapter

In this section, we identify the main problems that arise during all the phases of an ETL process. Then, we propose a modeling approach for the construction of ETL workflows, which is based on the life cycle of the ETL processes.

Problems and Issues of DW Refreshment

In all the phases of an ETL process (extraction and transportation, transformation and cleaning, and loading), individual issues arise, making data warehouse refreshment a very troublesome task. In the sequel, in order to clarify the complexity and the special characteristics of the ETL processes, we briefly review several issues, problems, and constraints that turn up in each phase separately.

- Global problems and constraints: Scalzo (2003) mentions that 90% of the problems in data warehouses arise during the loading of the data at the nightly batch cycles. At this period, the administrators have to deal with problems such as (a) efficient data loading and (b) concurrent job mixture and dependencies. Moreover, ETL processes have global time constraints including the initiation time and their completion deadlines. In fact, in most cases, there is a tight "time window" in the night that can be exploited for the refreshment of the data warehouse, since the source system is off-line or not heavily used during this period.

  Consequently, a major problem arises with the scheduling of the overall process. The administrator has to find the right execution order for dependent jobs and job sets on the existing hardware for the permitted time schedule. On the other hand, if the OLTP applications cannot produce the necessary source data in time for processing before the data warehouse comes online, the information in the data warehouse will be out of date. Still, since data warehouses are used for strategic purposes, this problem can sometimes be afforded, due to the fact that long-term reporting/planning is not severely affected by this type of failures.
- **Extraction and transportation:** During the ETL process, one of the very first tasks that must be performed is the extraction of the relevant information that has to be further propagated to the warehouse (Theodoratos, Ligoudistianos, & Sellis, 2001). In order to minimize the overall processing time, this involves only a fraction of the source data that has changed since the previous execution of the ETL process, mainly concerning the newly inserted and possibly updated records. Usually, change detection is physically performed by the comparison of two snapshots (one corresponding to the previous extraction and the other to the current one). Efficient algorithms exist for this task, like the snapshot differential algorithms presented by Labio and Garcia-Molina (1996). Another technique is log "sniffing," that is, the scanning of the log file in order to "reconstruct" the changes performed since the last scan. In rare cases, change detection can be facilitated by the use of triggers. However, this solution is technically impossible for many of the sources that are legacy systems (such a technique adds an enormous load to the source systems) or plain flat files.

In numerous other cases, where relational systems are used at the source side, the usage of triggers is also prohibitive both due to the performance degradation that their usage incurs and the need to intervene in the structure of the database. Moreover, another crucial issue concerns the transportation of data after the extraction, where tasks like FTP, encryption-decryption, compression-decompression, and so forth, can possibly take place.

- **Transformation and cleaning:** It is possible to determine typical tasks that take place during the transformation and cleaning phase of an ETL process. Rahm and Hai Do (2000) further detail this phase in the following tasks: (a) data analysis; (b) definition of transformation workflow and mapping rules; (c) verification; (d) transformation; and (e) backflow of cleaned data. In terms of the transformation tasks, we distinguish two main classes of problems (Lenzerini, 2002): (a) conflicts and problems at the *schema level* (e.g., naming and structural conflicts) and (b) *data level* transformations (i.e., at the instance level). The main problems with respect to the schema level are (a) *naming conflicts*, where the same name is used for different objects (homonyms) or different names are used for the same object (synonyms) and (b) *structural conflicts*, where one must deal with different representations of the same object in different sources. In addition, there are a lot of variations of data-level conflicts across sources: duplicated or contradicting records, different value representations (e.g., for marital status), different interpretation of the values (e.g., measurement units dollar vs. euro), different aggregation levels (e.g., sales per product vs. sales per product group), or reference to different points in time (e.g., current sales as of yesterday for a certain source vs. as of last week for another source). The list is enriched by low-level technical problems like data type conversions, applying format masks, assigning fields to a sequence number, substituting constants, setting values to NULL or DEFAULT based on a condition, or using simple SQL operators, for instance, UPPER, TRUNC, SUBSTR. The integration and transformation programs perform a wide variety of functions, such as reformatting, recalculating, modifying key structures, adding an element of time, identifying default values, supplying logic to choose between multiple sources, summarizing, merging data from multiple sources, and so forth.
- **Loading:** The final loading of the data warehouse has its own technical challenges. A major problem is the ability to discriminate between new and existing data at loading time. This problem arises when a set of records has to be classified to (a) the new rows that need to be appended to the warehouse and (b) rows that already exist in the data warehouse, but their value has changed and must be updated (e.g., with an UPDATE command). Modern ETL tools already provide mechanisms towards this problem, mostly through language predicates, for example, Oracle's MERGE command (Oracle, 2002). Also,

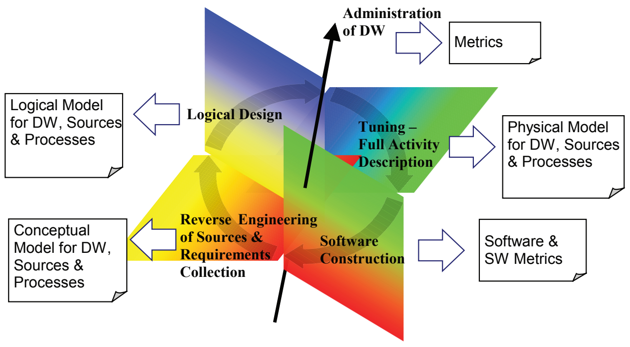
simple SQL commands are not sufficient since the open-loop-fetch technique, where records are inserted one by one, is extremely slow for the vast volume of data to be loaded in the warehouse. An extra problem is the simultaneous usage of the rollback segments and log files during the loading process. The option to turn them off contains some risk in the case of a loading failure. So far, the best technique seems to be the usage of the batch loading tools offered by most RDBMS that avoids these problems. Other techniques that facilitate the loading task involve the creation of tables at the same time with the creation of the respective indexes, the minimization of interprocess wait states, and the maximization of concurrent CPU usage.

Research Problems and Challenges

The previous discussion demonstrates the problem of designing an efficient, robust, and evolvable ETL workflow is relevant and pressing. To be more specific and understand the requirements of the design and evolution of a data warehouse, we have to clarify how ETL workflows fit in the data warehouse life cycle.

As we can see in Figure 2, the life cycle of a data warehouse begins with an initial reverse engineering and requirements collection phase where the data sources are analyzed in order to comprehend their structure and contents. At the same time, any requirements on the part of the users (normally a few power users) are also collected. The deliverable of this stage is a conceptual model for the data stores and the processes involved. In a second stage, namely the logical design of the warehouse, the logical schema for the warehouse and the processes are constructed. Third, the logical design of the schema and processes are optimized and refined to the choice

Figure 2. The life cycle of a data warehouse and its ETL processes (Vassiliadis, Simitsis & Skiadopoulos, 2002a)


Administration of DW
Metrics
Logical Model for DW, Sources & Processes
Logical Design
Tuning – Full Activity Description
Physical Model for DW, Sources & Processes
Conceptual Model for DW, Sources & Processes
Reverse Engineering of Sources & Requirements Collection
Software Construction
Software & SW Metrics

of specific physical structures in the warehouse (e.g., indexes) and environment-specific execution parameters for the operational processes. We call this stage *tuning* and its deliverable is the physical model of the environment. In a fourth stage, software construction, the software is constructed, tested, evaluated, and a first version of the warehouse is deployed. This process is guided through specific software metrics. Then, the cycle starts again, since data sources, user requirements, and the data warehouse state are under continuous evolution. An extra feature that comes into the scene after the deployment of the warehouse is the administration task, which also needs specific metrics for the maintenance and monitoring of the data warehouse. Consequently, in order to achieve our goal we have to deal with the phases of the life cycle of a data warehouse.

- **Conceptual model:** A conceptual model for ETL processes deals with the earliest stages of the data warehouse design. During this period, the data warehouse designer is concerned with two tasks which are practically executed in parallel: (a) the collection of requirements from the part of the users, and (b) the analysis of the structure and content of the existing data sources and their intentional mapping to the common data warehouse model. The design of an ETL process aims at the production of a crucial deliverable: the mapping of the attributes of the data sources to the attributes of the data warehouse tables through the appropriate intermediate transformations. The production of this deliverable involves several interviews that result in the revision and redefinition of original assumptions and mappings; thus it is imperative that a simple conceptual model should be employed in order to facilitate the smooth redefinition and revision efforts and to serve as the means of communication with the rest of the involved parties.

  From our point of view, a conceptual model for ETL processes shall not be another process/workflow model for the population of the data warehouse. There are two basic reasons for this approach. First, in the conceptual model for ETL processes, the focus is on documenting/formalizing the particularities of the data sources with respect to the data warehouse and not in providing a technical solution for the implementation of the process. Second, the ETL conceptual model is constructed in the early stages of the data warehouse project during which the time constraints of the project require a quick documentation of the involved data stores and their relationships, rather than an in-depth description of a composite workflow.
- **Logical model:** In the logical perspective, we classify the design artifacts that describe an abstraction of the workflow environment. First, the designer is responsible for defining an Execution Plan for the scenario. The definition of an execution plan can be seen from various views. The Execution Sequence involves the specification of (a) which process runs first, second, and so on;

(b) which processes run in parallel; or (c) when a semaphore is defined so that several processes are synchronized at a rendezvous point. ETL processes normally run in batches, so the designer needs to specify an Execution Schedule, that is, the time points or events that trigger the execution of the workflow as a whole. Finally, due to system crashes, it is imperative that a recovery plan exists, specifying the sequence of steps to be taken in the case of failure for a certain process (e.g., retry to execute the process, or undo any intermediate results produced so far). In the ETL case, due to the data centric nature of the process, the designer must deal with the relationship of the involved processes with the underlying data. This involves the definition of a primary data flow that describes the route of data from the sources towards their final destination in the data warehouse, as they pass through the processes of the workflow. Also, due to possible quality problems of the processed data, the designer is obliged to define a data flow for logical exceptions, which is responsible for the flow of the problematic data, that is, the rows that violate integrity or business rules. Moreover, a very crucial topic is the semantics of the ETL workflow. These semantics are generated by the combination of the data flow and the execution sequence: the data flow defines what each process does and the execution plan defines in which order and combination.

- **Mapping conceptual to logical models:** Another issue that has to be solved is the transition between the aforementioned phases (i.e., conceptual and logical) of the data warehouse life cycle. On one hand, there exists a simple model, sufficient for the early stages of the data warehouse design. On the other hand, there exists a logical model that offers formal and semantically founded concepts to capture the particularities of an ETL process.

  The goal of this transition should be to facilitate the integration of the results accumulated in the early phases of a data warehouse project into the logical model, such as the collection of requirements from the part of the users, the analysis of the structure and content of the existing data sources, along with their intentional mapping to the common data warehouse model. The deliverable of this transition is not expected to be always a complete and accurate logical design. The designer/administrator in the logical level should examine, complement, or change the outcome of this methodology, in order to achieve the goals.

  In the context of finding an automatic transition from one model to the other, there are several problems that should be addressed. Since the conceptual model is constructed in a more generic and high-level manner, each conceptual entity has a mapping to a logical entity. Thus, there is a need for the determination of these mappings. Moreover, we have stressed that the conceptual model is not a workflow as it simply identifies the transformations needed in an ETL process. Therefore, it does not directly specify the execution order of these transformations. On the other hand, the execution order is a very important property of the logical model. So, there is a necessity for finding a way to specify the execution order of the transformations in an ETL process during the transition between the two models.

- **Optimization of ETL workflows:** In order to design an efficient, robust, and evolvable ETL workflow, we have to optimize its execution plan. In other words, we have to optimize the sequence of the ETL operations involved in the overall process.

  Up to now, the research community has confronted the problem of the optimization of data warehouse refreshment as a problem of finding the optimal strategy for view maintenance. But this is not sufficient with respect to mechanisms that are employed in real-world settings. In fact, in real-world data warehouse environments, this procedure differs to the point that the execution of operational processes (which is employed in order to export data from operational data sources, transform them into the format of the target tables, and finally, load them to the data warehouse) does not like as a "big" query; rather it is more realistic to be considered a complex transaction. Thus, there is a necessity to deal with this problem for a different perspective by taking into consideration the characteristics of an ETL process presented in the previous subsection. One could argue that we can possibly express all ETL operations in terms of relational algebra and then optimize the resulting expression as usual. But, the traditional logic-based algebraic query optimization can be blocked, basically due to the existence of data manipulation functions.

  However, if we study the problem of the optimization of ETL workflows from its logical point of view, we can identify several interesting research problems and optimization opportunities. At first, there is a necessity for a framework that will allow the application of several well-known query optimization techniques to the optimization of ETL workflows. For example, it is desirable to push selections all the way to the sources, in order to avoid processing unnecessary rows. Moreover, it is desirable to determine appropriate techniques and requirements, so that an ETL transformation (e.g., a filter) can be pushed before or after another transformation involving a function. Additionally, there is a need to tackle the problem of homonyms. For example, assume the case of two attributes with the same name, COST, where the first one has values in European currency, while the other contains values in American currency. Clearly, in this case it is not obvious if a transformation that involves the first attribute (e.g., a transformation that converts the values from American to European currency) can be pushed before or after another transformation that involves the second attribute (e.g., a transformation that filters the values over a certain threshold).

- **Software construction:** To conclude the discussion about the life cycle of the data warehouse presented in Figure 2, one anticipates that the outcome of the aforementioned analysis should be used for the construction of a software prototype. This construction phase includes the development, testing, and deployment of a first version of the data warehouse. This process has to be guided through specific metrics for the maintenance and monitoring of the data warehouse.

A Roadmap for a Data Warehouse Designer/Administrator

We summarize the previously mentioned issues in Table 1, where we present a set of steps for the data warehouse designers towards constructing the backstage of the data warehouse. We organize the tasks of the designers in four phases (requirements, design, tuning, and implementation) of the software project. For each phase, we present the issues and the steps to follow concerning the main tasks of the warehouse architecture (extraction, transformation, cleaning). The fundamental deliverable that guides this process is also listed.

Arktos II: A Framework towards the Modeling and the Optimization of ETL Workflows

In this subsection, we present a framework for traditional data warehouse flows, named Arktos II, as an exemplary ETL modeling methodology that follows the life cycle of a data warehouse as we previously presented it. Below, we briefly present the main contributions of Arktos II, grouped by the phases of the life cycle of a data warehouse.

Arktos II (Simitsis, Vassiliadis, & Sellis, 2005a; Vassiliadis, Simitsis, Georgantas, Terrovitis, & Skiadopoulos, 2005) is a framework that studies the design, development, and optimization of ETL workflows. The uttermost goal of this framework is to facilitate, manage, and optimize the design and implementation of the ETL workflows during the initial design and deployment stage and during the continuous evolution of the data warehouse. Despite the fact that its prototype is only a design environment, at least for the moment, it benefits compared to the commercial ETL tools due to the logical abstraction that it offers; on the contrary, commercial tools are concerned directly with the physical perspective of an ETL scenario (at least to the best of our knowledge).

Arktos II proposes a novel conceptual model for the early stages of a data warehouse project (Vassiliadis, Simitsis, & Skiadopoulos, 2002). This model focuses on (a) the interrelationships of attributes and concepts and (b) the necessary transformations that need to take place during the loading of the warehouse. Also, it is able to capture constraints and transformation composition. Due to the nature of the design process, the features of the conceptual model are presented in a set of design steps that constitute a methodology for the design of the conceptual part of the overall ETL process. The construction of the model is realized in a customizable and extensible manner, so that the designer can enrich it with the designer's own reoccurring patterns for ETL transformations. Furthermore, it is enriched with a "palette" of frequently used ETL transformations, like the assignment of surrogate keys, the check for null values, and so on.

Additionally, Arktos II presents a formal logical model for the ETL environment that concentrates on the flow of data from the sources towards the data warehouse through the composition of activities and data stores (Vassiliadis et al., 2002). The flow of data from producers towards their consumers is achieved through the usage of provider relationships that map the attributes of the former to the respective attributes of the latter. A serializable combination of ETL activities, provider relationships, and data stores constitutes an ETL workflow. Also, a reusability framework that complements the genericity and customization of the metamodel is provided. Finally, Arktos II introduces techniques for the measurement of ETL workflows.

Furthermore, Arktos II proposes a semiautomatic transition from conceptual to logical model for ETL processes (Simitsis, 2005). The constituents of the conceptual model are mapped to their respective constituents of the logical model. Also, it presents a method for the determination of a correct execution order of the activities in the logical model, wherever feasible, by grouping the transformations of the conceptual design into stages.

Moreover, Arktos II delves into the logical optimization of ETL workflows, having as its uttermost goal the finding of the optimal ETL workflow (Simitsis, Vassiliadis, & Sellis, 2005). The method proposed reduces the execution cost of an ETL workflow, by changing either the total number or the execution order of the processes. The problem is modeled as a state space search problem, with each state representing a particular design of the workflow as a graph. The tuning of an ETL workflow is realized through several algorithms for the optimization of the execution order of the activities.

Finally, to replenish the aforementioned issues, an ETL tool has prototypically been implemented with the goal of facilitating the design, the (re)use, and the optimization of ETL workflows. The general architecture of Arktos II comprises a GUI, an ETL library, a metadata repository, and an optimizer engine. The GUI facilitates the design of ETL workflows in both the conceptual and logical level, through a workflow editor and a template palette. The ETL library contains template code of built-in functions and maintains a template code of user-defined functions. After its creation, the ETL workflow is propagated to the optimizer in order to achieve a better version with respect to the execution time. All the aforementioned components are communicating with each other through the metadata repository.

Future Trends

In our opinion, there are several issues that are technologically open and present interesting topics of research for the future in the field of data integration in data warehouse environments. This opinion is supported by the results of a recent workshop located in Dagstuhl, Germany in the Summer of 2004 (Dagstuhl Perspective Workshop, 2004), where several researchers tried to articulate the most pressing issues for the next directions of data warehousing research. A research agenda describing opportunities and challenges for promising new areas in data warehousing research was proposed that focuses on architecture, processes, modeling and design, and novel applications. Out of these, the participants discuss the future role of ETL in a threefold categorization: traditional ETL, stream ETL, and on-demand ETL. We adopt this classification, and for each one of these categories, we present a list of several future directions.

Table 1. Typical tasks for a data warehouse designer/developer organized by phase and stage

	Extract	**Transform**	**Load**	**Deliverable**
Phase 1a: Reverse Engineering of Sources & Requirements Collection	▪ collection of requirements from the part of the users ▪ analysis of the structure and content of the existing data sources and their intentional mapping to the common data warehouse model	▪ mapping of the attributes of the data sources to the attributes of the data warehouse tables through the appropriate intermediate transformations	▪ identification of data targets	▪ conceptual schema
Phase 1b: Define Sources and Processes	▪ concepts map to logical recordsets	▪ transformations map to logical activities ▪ definition of a proper execution order	▪ concepts map to logical recordsets ▪ ETL constraints map to logical activities	▪ a set of steps for the transition of conceptual to logical schemas
Phase 2: Logical Design	▪ definition of a plan for the population of the Data Staging Area w.r.t. schemata mappings ▪ choice of extraction policy: incremental or full extraction	▪ description of ETL activities w.r.t. schemata mappings and internal semantics ▪ identification of an optimal execution plan	▪ appropriate schemata mappings	▪ logical schema

Table 1. continued

Phase 3: Tuning—Full Activity Description	▪ population of the Data Staging Area w.r.t. security and network issues ▪ define a schedule for extraction jobs (e.g., during the night)	▪ physical optimization: choose an appropriate algorithm/code for each activity involved in the whole ETL workflow ▪ re-evaluate the execution plan w.r.t. the different alternatives for each activity (e.g., merge-join vs. nested-loops)	▪ choice of a load policy: e.g., bulk loading, creation of indexes, minimization of interprocess wait states, maximization of concurrent CPU usage	▪ physical schema
Phase 4: Software Construction	▪ resolution of connection issues (DNS, SID, etc.) ▪ implementation of the appropriate extraction scripts ▪ testing	▪ implementation of the mentioned algorithms/codes for each activity in the ETL workflow ▪ testing	▪ implementation of the appropriate loading scripts ▪ testing	▪ software artifact

- **Traditional ETL:** Traditional ETL processes are responsible for the extraction of data from several sources, their cleansing, customization, and insertion into a data warehouse. These tasks are repeated on a regular basis, and in most cases, they are asynchronous. As research challenges in this area, we mention the following issues:
    - A formal description of ETL processes with particular emphasis on an algebra (for optimization purposes) and a formal declarative language.
    - The optimization of ETL processes on logical and physical levels. A challenge will be either the optimization of the whole ETL process or of any individual transformation. Parallel processing of ETL processes is of particular importance.
    - The propagation of changes back to the sources. Potential quality problems observed at the end-user level can lead to clean data being propagated back to the sources, in order to avoid the repetition of several tasks in future application of the ETL process. Clearly, this idea has already been mentioned in the literature as "backflow of cleaned data" (Rahm & Hai Do, 2000), but the problem is not solved yet.
    - The provision of standard-based metadata for ETL processes. There does not exist common model for the metadata of ETL processes. CWM is not sufficient for this purpose and it is too complicated for real-world applications.

- ○ The integration of ETL with XML adapters, EAI (Enterprise Application Integration) tools (e.g., MQ-Series), and data quality tools.
- ○ The extension of the ETL mechanisms for nontraditional data, like XML/HTML, spatial, and biomedical data.
- ○ The treatment of security issues in ETL; source data and data in transit are security risks (Friedman, 2002a).

- **Stream ETL:** Streams are sequences of data, continuously flowing from a data source with the particular characteristic that, due to their volume, each tuple is available only for a limited time window for querying. Stream examples would involve stock rates extracted from the Web, packets going through a router, clickstreams from a Web site, and so forth. Stream ETL is an ETL process involving the possible filtering, value conversion, and transformations of this incoming information in a relational format. Although, streams cannot be stored, some patterns or snapshot aggregates of them can be stored for subsequent querying. As research challenges in this area, we can mention the following issues:
    - ○ The necessity of maintaining data in the DW as much "online" as we can, but without adding an extra load to sources or DW.
    - ○ The provision of correctness guarantees.
    - ○ The necessity of cost models for the tuning of the incoming stream within specified time window.
    - ○ The audit of the incoming stream data for several constraints or business rules, also with respect to stored data (e.g., primary key violations).
- **On-Demand ETL:** An ETL process of this kind is executed sporadically, and it is manually initiated by some user demand. The process is responsible for retrieving external data and loading them in the DW after the appropriate transformations. For instance, consider the case that some users request data to be brought in from the Web. The administrator/programmer is assigned the task of constructing an ETL process that extracts the dates from the specified sites, transforms them, and ultimately stores them in some (possibly novel) part of the warehouse. Any time the user needs this data, this on-demand ETL process brings in the relevant information. As research challenges in this area, we mention the following issues:
    - ○ The need for appropriate operators, since this process is mostly focused towards Web data.
    - ○ The computation of minimum effort/time/resources for the construction of the process.
    - ○ The provision of a framework easily adaptable to the changes of the external data.

- The finding of efficient algorithms, due to the fact that this process is initiated by the user.

Conclusion

In this chapter, we have delved into a crucial part of the data warehouse architecture: the backstage area. We have presented the state of the art concerning the existing ETL technology. In practice, a designer/administrator uses a commercial ETL tool or an in-house developed software artifact to create ad-hoc ETL workflows. We have stressed the fact that there is not a unified approach that concretely deals with the modeling and the optimization of ETL workflows.

Additionally, we have indicated the main challenges and problems concerning the manufacturing of ETL workflows. In all the phases of an ETL process (extraction and transportation, transformation and cleaning, and loading) individual issues arise, making data warehouse refreshment a very troublesome task. In general, ETL workflows are characterized as quite complex, costly, critical, and important for the success of a data warehouse project. The key factors underlying the main problems of ETL workflows are vastness of the data volumes, quality problems, performance, evolution of the sources, and the data warehouse.

Moreover, we have presented a modeling approach for the construction of ETL workflows, which is based on the life cycle of the ETL processes. This life cycle consist of four phases: reverse engineering and requirements collection, logical design, tuning and physical design, and software construction. As a result, in order to construct an efficient, robust, and evolvable ETL workflow, we have to deal with all the phases of the life cycle of a data warehouse.

Finally, we have pointed out a list of open research issues arranged in three basic categories: traditional ETL, stream ETL, and on-demand ETL; and we have provided several future directions.

References

Adzic, J., & Fiore, V. (2003, September). Data warehouse population platform. In *Proceedings of 5th International Workshop on the Design and Management of Data Warehouses (DMDW'03),* Berlin, Germany.

Ascential. (2003). *Data warehousing technology.* Retrieved May 27, 2006, from http://www.ascentialsoftware.com/products/datastage.html

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide.* Addison-Wesley.

Borkar, V., Deshmuk, K., & Sarawagi, S. (2000). Automatically extracting structure from free text addresses. *Bulletin of the Technical Committee on Data Engineering, 23*(4).

Bouzeghoub, M., Fabret, F., & Matulovic, M. (1999). Modeling data warehouse refreshment process as a workflow application. In *Proceedings of 1st International Workshop on the Design and Management of Data Warehouses (DMDW'99),* Heidelberg, Germany.

Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M., Naggar, P., & Vernacotola, F. (2003). IBIS: Semantic data integration at work. In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, Klangefurt, Austria (pp. 79-94).

Cohen, W. (1999). Some practical observations on integration of Web information. In *Proceedings of SIGMOD Workshop on the Web and Databases (WebDB'99)*, Philadelphia, Pennsylvania.

Dagstuhl Perspective Workshop. (2004). *Data warehousing at the crossroads.* Dagstuhl, Germany.

Demarest, M. (1999). *The politics of data warehousing.* Retrieved May 27, 2006, from http://www.hevanet.com/demarest/marc/dwpol.html

Eder, J., & Gruber, W. (2002). A meta model for structured workflows supporting workflow transformations. In *Proceedings of the 6th East European Conference on Advances in Databases and Information Systems (ADBIS'02)*, Bratislava, Slovakia (pp. 326-339).

Friedman, T. (2002). *Reducing the cost of ETL for the data warehouse* (Tech. Rep. No. COM-16-8237). Gartner Group.

Friedman, T. (2002a). *Security issues in ETL for the data warehouse* (Tech. Rep. No. COM-17-8459). Gartner Group.

Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (1999). *An extensible framework for data cleaning* (Tech. Rep. No. INRIA RR-3742).

Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (2000). Ajax: An extensible data cleaning tool. In *Proceedings of ACM International Conference on the Management of Data (SIGMOD'00)*, Dallas, Texas (p. 590).

Gartner. (2003). *ETL Magic Quadrant update: Market pressure increases* (Gartner's Strategic Data Management Research Note M-19-1108). Author.

Giga. (2002). *Market overview update: ETL* (Tech. Rep. No. RPA-032002-00021). Author.

IBM. (2003). *IBM Data Warehouse Manager.* Retrieved May 27, 2006, from http://www-3.ibm.com/software/data/db2/datawarehouse

Informatica. (2003). *PowerCenter.* Retrieved May 27, 2006, from http://www.informatica.com/products/data+integration/powercenter/default.htm

Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (Eds.) (2003). *Fundamentals of data warehouses* (2nd ed.). Germany: Springer-Verlag.

Jarke, M., List, T., & Koller, J. (2000). The challenge of process warehousing. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, Cairo, Egypt.

Jarke, M., Quix, C., Blees, G., Lehmann, D., Michalk, G., & Stierl, S. (1999). Improving OLTP data quality using data warehouse mechanisms. In *Proceedings of ACM International Conference on Management of Data (SIGMOD'99)*, Philadelphia, Pennsylvania (pp. 537-538).

Labio, W., & Garcia-Molina, H. (1996). Efficient snapshot differential algorithms for data warehousing. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, Bombay, India (pp. 63-74).

Kiepuszewski, B., Ter Hofstede, A. H. M., & Bussler, C. (2000). On structured workflow modeling. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00)*, Stockholm, Sweden (pp. 431-445).

Lenzerini. (2002). Data integration: A theoretical perspective. In *Proceedings of the 21st Symposium on Principles of Database Systems (PODS'02)*, Wisconsin (pp. 233-246).

Microsoft. (2003). *Data transformation services.* Retrieved May 27, 2006, from http://www.microsoft.com

Monge, A. (2000). Matching algorithms within a duplicate detection system. *Bulletin of the Technical Committee on Data Engineering, 23*(4).

Oracle. (2001). Oracle9i™ Warehouse Builder user's guide (Release 9.0.2). Retrieved May 27, 2006, from http://otn.oracle.com/products/warehouse/content.html

Oracle. (2002). *Oracle9i™ SQL reference* (Release 9.2, pp. 17.77-17.80). Author.

Rahm, E., & Hai Do, H. (2000). Data cleaning: Problems and current approaches. *Bulletin of the Technical Committee on Data Engineering, 23*(4).

Raman, V., & Hellerstein, J. (2001). Potter's Wheel: An interactive data cleaning system. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, Roma, Italy (pp. 381-390).

Rundensteiner, E. (Ed.). (1999). Special issue on data transformations. *Bulletin of the Technical Committee on Data Engineering, 22*(1).

Sadiq, W., & Orlowska, M. E. (2000). On business process model transformations. In *Proceedings of the 19th International Conference on Conceptual Modeling (ER'00)* (pp. 267-280), Salt Lake City, Utah.

Sarawagi, S. (2000, December). Special issue on data cleaning. *Bulletin of the Technical Committee on Data Engineering, 23*(4).

Scalzo, B. (2003). *Oracle DBA guide to data warehousing and star schemas.* Prentice Hall.

Schafer, E., Becker, J.-D., & Jarke, M. (2000). DB-Prism: Integrated data warehouses and knowledge networks for bank controlling. In *Proceedings of the 26th International Conference on Very Large Databases*, Cairo, Egypt *(VLDB'00)*.

Shilakes, C., & Tylman, J. (1998). Enterprise information portals. Retrieved May 27, 2006, from http://www.sagemaker.com/company/downloads/eip/indepth.pdf

Simitsis, A. (2004). *Modeling and optimization of extraction-transformation-loading (ETL) processes in data warehouse environments.* Doctoral Thesis, NTU Athens, Greece.

Simitsis, A. (2005). Mapping conceptual to logical models for ETL processes. In *Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP (DOLAP '05)*, Bremen, Germany.

Simitsis, A., Vassiliadis, P., & Sellis, T. (2005). Optimizing ETL processes in data warehouse environments. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE '05)*, Tokyo, Japan.

Simitsis, A., Vassiliadis, P., & Sellis, T. (2005a). State-space optimization of ETL workflows. *IEEE Transactions on Knowledge and Data Engineering, 17*(10), 1404-1419.

Strange, K. (2002). *ETL was the key to this data warehouse's success* (Tech. Rep. No. CS-15-3143). Gartner.

Strange, K. (2002a). *Data warehouse TCO: Don't underestimate the cost of ETL* (Tech. Rep. No. DF-15-2007). Gartner Group.

Theodoratos, D., Ligoudistianos, S., & Sellis, T. (2001). View selection for designing the global data warehouse. *Data & Knowledge Engineering, 39*(3), 219-240.

Trujillo, J., & Luján-Mora, S. (2003). A UML based approach for modeling ETL processes in data warehouses. In *Proceedings of 22nd International Conference on Conceptual Modeling (ER 2003)*, Chicago, Illinois (pp. 307-320). LNCS, 2813.

Van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2000). *Workflow patterns* (BETA Working Paper Series WP 47). Eindhoven University of Technology, Eindhoven.

Vassiliadis, P. (2000). Gulliver in the land of data warehousing: Practical experiences and observations of a researcher. In *Proceedings of 2nd International Workshop on Design and Management of Data Warehouses (DMDW'00)*, Stockholm, Sweden.

Vassiliadis, P., Quix, C., Vassiliou, Y., & Jarke, M. (2001). Data warehouse process management. *Information Systems, 26*(3), 205-236.

Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., & Skiadopoulos, S. (2005). A generic and customizable framework for the design of ETL scenarios. *Information Systems, 30*(7), 492-525.

Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. In *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP (DOLAP '02)*, McLean, Virginia.

Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002a). Modeling ETL activities as graphs. In *Proceedings of the 4th International Workshop on the Design and Management of Data Warehouses (DMDW '02)*, Toronto, Canada (pp. 52-61).

Workflow Management Coalition. (1998). *Interface 1: Process definition interchange process model* (Doc. No. WfMC TC-1016-P). Author.

Section III

Efficiency of Analytical Processing