1. Introduction This chapter highlights the background of the dissertation and outlines its structure. In Section 1.1 we briefly present a high-level architecture of a data warehouse and we exhibit its main layers. In Section 1.2 we introduce the Extract – Transform – Load (ETL) processes, the general subject of this work. In Section 1.3, we sketch our approach for modeling ETL processes, and formulate our main objectives. An overview of the structure and contributions of the dissertation is given in Section 1.4. 1.1 Data Warehouses The term data warehouse, in a high-level description, stands for a collection of technologies that aims at enabling the knowledge worker (executive, manager, analyst, etc.) to make better and faster decisions by exploiting integrated information of the different information systems of his/her organization. The most popular definition of data warehouses is found in [Inmo96]: ‘A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data used to support the strategic decision-making process for the enterprise. It is the central point of data integration for business intelligence and is the source of data for the data marts, delivering a common view of enterprise data’. Almost a decade of research has been spent on the study of data warehouses, especially for its design and exploitation for decision-making purposes. Data warehouses are typically composed of the front stage, concerning end-users who access the data warehouse with decision support tools, and the back stage, where administrators undertake the task to populate the data warehouse with data obtained from several sources. The architecture of a data warehouse exhibits various layers in which data from one layer are derived from data of the previous layer (Figure 1.1). The Data Sources, also called operational databases, form the starting layer. Data sources may consist of structured data stored in On-Line Transaction Processing (OLTP) database and legacy systems or even in files. The next layer comprises the back stage part of a data warehouse, where the collection, integration, cleaning and transformation of data take place in order to populate the warehouse. An auxiliary area of volatile data, called Data Staging Area (DSA), is employed for the purpose of data transformation, reconciliation and cleaning. The central layer of the architecture is the global Data Warehouse (DW). The global data warehouse keeps a historical record of data that result from the transformation, integration, and aggregation of detailed data found in the data sources. Moreover, this layer involves client warehouses, which contain 1. Introduction 2 highly aggregated data, directly derived from the global warehouse. There are various kinds of client warehouses, such as the Data Marts or the On-Line Analytical Processing (OLAP) databases, which may use relational database systems or specific multidimensional data structures. The whole environment is described in terms of its components, metadata and processes in a central Metadata Repository, located at the data warehouse site [JLVV03]. Figure 1.1. Abstract architecture of a Data Warehouse The front-end level of the data warehouse architecture consists of applications and techniques that business users use to interact with data stored in the data warehouse. Data mining techniques are used for discovery of data patterns, for prediction analysis and for classification. OLAP techniques are used for a specialized style of querying that exploits the usual (multi-)dimensional design of a data warehouse, and present text and number data from data warehouses, in order to analyze historical data and slice the business information required. Reporting tools are used to generate reports on the data. Data visualization tools are used to formulate and show data according to business rules and/or the results of data mining and OLAP techniques. 1.2 Data Warehouse Operational Processes 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 simply insufficient to describe the structure and contents of a data warehouse in realworld settings. In [VQVJ01], the authors 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 ones 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 1.2 Data Warehouse Operational Processes 3 goal, we have to (a) specify data transformations as a workflow; (b) optimize this workflow; and (c) execute the data transformation workflow. Data warehouse operational processes normally compose a labor intensive workflow and constitute an integral part of the back-stage of data warehouse architectures, where the collection, extraction, cleaning, transformation, and transport of data takes place, in order to populate the warehouse. To deal with this workflow and in order to facilitate and manage the data warehouse operational processes, specialized tools are already available in the market (see Section 2.1 for a broader discussion), under the general title extraction-transformation-loading (ETL) tools. Extraction-Transformation-Loading (ETL) tools are pieces of software responsible for the extraction of data from several sources, their cleansing, their customization, their transformation in order to fit business needs, and finally, their loading into a data warehouse. 1.2.1 Functionality of ETL Tools ETL tools represent an important part of data warehousing, as they represent the mean in which data actually gets loaded into the warehouse. To give a general idea of the functionality of these tools we mention their most prominent tasks, which include: (a) the identification of relevant information at the source side, (b) the extraction of this information, (c) the transportation of this information to the DSA, (d) the transformation, (i.e., customization and integration) of the information coming from multiple sources into a common format, (e) the cleaning of the resulting data set, on the basis of database and business rules, and (f) the propagation and loading of the data to the data warehouse and the refreshment of data marts. 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. Figure 1.2. The environment of Extraction-Transformation-Loading processes In Figure 1.2, we abstractly describe the general framework for ETL processes. In 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 1. Introduction 4 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 1.2 and comprises the target data stores, i.e., fact tables for the storage of information and dimension tables with the description and the multidimensional, roll-up 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. 1.2.2 Phases 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 some of the peculiarities of ETL processes, we briefly review several issues, problems, and constraints that turn up in each phase separately. Global problems and constraints. [Scal03] mentions that 90% of the problems in data warehouses arise from the nightly batch cycles that load the data. At this period, the administrators have to deal with problems such (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 & 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 [ThLS01]. 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 in [LaGa96]. Another technique is log ‘sniffing’, i.e., 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 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 1.2 Data Warehouse Operational Processes 5 of data after the extraction, where tasks like ftp, encryption - decryption, compression - decompression, etc., can possibly take place. Transformation & cleaning. It is possible to determine typical tasks that take place during the transformation and cleaning phase of an ETL process. [RaDo00] further details 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 [Lenz02]: (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; e.g., UPPER, TRUNC, SUBSTR, etc. 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, etc. 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 (e.g., Oracle’s MERGE command [Orac02]). 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’s 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 inter-process wait states, and the maximization of concurrent CPU usage. 1. Introduction 6 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 inter-process wait states, and the maximization of concurrent CPU usage. 1.3 Issues Around the Modeling and Optimization of ETL Processes 1.3.1 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 [BoRJ98] 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 activities 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 in the market the last few years increased their sales from 101 US million dollars in 1998 to 210 US million dollars in 2002 having a steady increase rate of approximately 20.1% each year [JLVV03]. 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 the 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 [Dema97, ShTy98, Vass00]. 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 [Stra02, Giga02]. Complementary reports [Frie02, Stra02a] address the factors that influence the cost of its implementation and support: (a) staff (development, application support teams, ongoing 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, [Stra02] 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 percent of the overall IT staffing) – 55 from ETL, four database administrators, five architects, four systems administrators, nine BI competency center workers (assisting end users), five report writers, nine managers, and nine hardware, operating system and operations support staff members. These 55 individuals were responsible for building and maintaining the ETL process, which includes 46 different 1.3 Issues Around the Modeling and Optimization of ETL Processes 7 source systems. Responsibilities include updates of data marts on a weekly and monthly basis. This does not include staffing 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 4 terabytes or more of disk storage, at an acquisition cost over three years of $5 million. The cost of the ETL tool used was $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. As mentioned in [BoFM99] 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, [AdFi03] 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, [Stra02a] stress that the complexity of the ETL process, as well as the staffing required to implement it, depends on the following variables, among others: (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 above, 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, even to daily maintenance operations Visualizing and understanding this kind of systems is another issue. In fact, traditional modeling approaches need to be reconsidered: we need interactive, multi-view 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? 1. Introduction 8 - Is variant A better than variant B? Under these conditions, the problem of designing an efficient, robust and evolvable ETL workflow is relevant and pressing. Thus, we believe that the research on ETL processes is a valid research goal. 1.3.2 Research Problems and Challenges To probe into the issues presented in the previous subsection and understand the requirements of the design and evolution of a data warehouse, we have to clarify how ETL processes fit in the data warehouse lifecycle. As we can see in Figure 1.3, the lifecycle 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 from 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 is constructed. Third, the logical design of the schema and processes is optimized and refined to the choice 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, 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. Conceptual Model for DW, Sources & Processes Logical Design Tuning – Full Activity Description Software Construction Administration of DW Reverse Engineering of Sources & Requirements Collection Software & SW Metrics Physical Model for DW, Sources & Processes Logical Model for DW, Sources & Processes Metrics Figure 1.3. The lifecycle of a Data Warehouse and its ETL processes Consequently, in order to achieve our goal we have to deal with the phases of the lifecycle 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 1.3 Issues Around the Modeling and Optimization of ETL Processes 9 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. 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 is 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. We stress that, 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. Secondly, 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. We believe that the formal modeling of the starting concepts of a data warehouse design process has not been adequately dealt by the research community. This belief is fully justified in Chapter 2, where the state of the art is presented. Logical Model. If we treat an ETL scenario as a composite workflow, in a traditional way, its designer is obliged to define several of its parameters (Figure 1.4). We follow a multi-perspective approach that enables to separate these parameters and study them in a principled approach. Figure 1.4. Different perspectives for an ETL workflow 1. Introduction 10 We discriminate two parts in the lifecycle of the overall ETL process: administration and design, and we depict them at the lower and upper part of Figure 1.4, respectively. At the lower part of Figure 1.4, we are dealing with the tasks that concern the administration of the workflow environment and their dynamic behavior at runtime. First, an Administration Plan should be specified, involving the notification of the administrator either on-line (monitoring) or off-line (logging) for the status of an executed process, as well as the security and authentication management for the ETL environment. At the top of Figure 1.4, we depict the static design artifacts for a workflow environment. We follow a traditional approach and group the design artifacts into physical and logical, with each category comprising its own perspective. The physical perspective is depicted on the right hand side of Figure 1.4, and the logical perspective is depicted on the left hand side. On the right hand side of Figure 1.4, the physical perspective is presented and involves the registration of the actual entities that exist in the real world. We reuse the terminology of [AHKB00] for the physical perspective. The Resource Layer comprises the definition of roles (human or software) that are responsible for executing the processes of the workflow. The Operational Layer, at the same time, comprises the software modules that implement the design entities of the logical perspective in the real world. In other words, the processes defined at the logical layer (in an abstract way) are materialized and executed through the specific software modules of the physical perspective. This dissertation is mainly interested in the logical point of view. At the logical perspective, we classify the design artifacts that give an abstract description 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 which process runs first, second, and so on, which processes run in parallel or when a semaphore is defined so that several processes are synchronized at a rendezvous point. ETL processes normally run in batch, so the designer needs to specify an Execution Schedule, i.e., the time points or events that trigger the execution of the workflow as a whole. Finally, due to system crashes, it is imperative that there exists a Recovery Plan, 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). We find that research has not dealt with the definition of data-centric workflows to the entirety of its extent. 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, i.e., the rows that violate integrity or business rules. In our work, we are interested in 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 lifecycle. On one hand, there exists a simple model, sufficient for the early stages of the data warehouse design. On the other 1.3 Issues Around the Modeling and Optimization of ETL Processes 11 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 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 his/her 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 highlevel 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. We have already pointed out the significance of the problem of designing an efficient, robust and evolvable ETL workflow. In order to design such a 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, 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 as we have already stressed, 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 as 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. Actually, as we discuss in detail in the next chapter, research has only partially dealt with the problem of designing and managing ETL workflows. To our knowledge, there is no systematic treatment of the problem yet, as far as the problem of the design of an optimal ETL workflow is concerned. 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 1. Introduction 12 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 lifecycle of the data warehouse presented in Figure 1.3, 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 the 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. After presenting the motivation along with research problems and challenges that should concern the study of the ETL processes, we move to discuss our approach. 1.4 Contributions and Outline of the Dissertation Our dissertation presents a framework towards the modeling of ETL processes and the optimization of ETL workflows. The uttermost goal of our research is to facilitate, manage and optimize the design and implementation of the ETL processes both during the initial design and deployment stage and during the continuous evolution of the data warehouse. Next, we discuss the contributions of our research, grouped by the phases of the lifecycle of a data warehouse presented in the previous subsection. Novelty of our Approach. We have reviewed the state of the art and practice for ETL tools, and we have come to the conclusion that the formal modeling and the optimization of ETL workflows have not been adequately dealt by the research community. In our review, at first, we present standards and the leading commercial ETL tools. Then, we discuss related work in the fields of conceptual modeling for data warehousing as well as for the field of ETL. Next, we refer to research efforts and solutions given in the academic community concerning research prototypes, transformations, data cleaning, and applications of workflow technology Moreover, we present the state of the art concerning the correlation of two different levels of design: conceptual and logical. Finally, we review research efforts in the field of optimization of ETL processes. The results of our study are presented in Chapter 2. 1.4 Contributions and Outline of the Dissertation 13 Conceptual Model. In order to tackle the problems presented in the previous subsection concerning the modeling of the early stages of a data warehouse project we propose a novel conceptual model, with a particular focus on (a) the interrelationships of attributes and concepts, and (b) the necessary transformations that need to take place during the loading of the warehouse. The former are directly captured in the proposed metamodel as a first class citizens. Attribute interrelationships are captured through provider relationships that map data provider attributes at the sources to data consumers in the warehouse. Apart from these fundamental relationships, the proposed model 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, which lead to the basic target; i.e., the attribute interrelationships. These steps constitute the 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 his own reoccurring patterns for ETL activities. Also, the model is enriched with a ‘palette’ of frequently used ETL activities, like the assignment of surrogate keys, the check for null values, etc. Our results on these topics are presented in Chapter 3. Previously, they have also been published in [SiVa03, VaSS02]. Logical Model. In our research, we have dealt with the definition of data-centric workflows. To achieve this, we define a formal logical metamodel as a logical abstraction of ETL processes. The data stores, activities and their constituent parts are formally defined. An activity is defined as an entity with (possibly more than one) input schema(ta), an output schema and a parameter schema, so that the activity is populated each time with its proper parameter values. 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. Moreover, the aforementioned model is reduced to a graph, which is called the Architecture Graph. All the entities are modeled as nodes and four different kinds of relationships are modeled as edges. These relationships involve: (a) type checking information (i.e., which type an entity corresponds to); (b) part-of relationships (e.g., which activity does an attribute belong to); (c) regulator relationships, covering the population of the parameters of the activities from attributes or constant values, and (d) provider relationships, covering the flow of data from providers to consumers. Also, a reusability framework that complements the genericity of the metamodel is provided. Practically, this is achieved by employing an extensible set of specializations of the entities of the metamodel layer. We employ a declarative database programming language, LDL, to define the semantics of each template activity. In fact, due to language considerations, two categories of template activities are offered under which the designer can choose to register his/hers template activity. Finally, we exploit our modeling to introduce techniques for the measurement of ETL workflows. Furthermore, we discuss the usage and intuition of these measures for the maintenance and what-if analysis of the ETL workflow. Our results are presented in Chapter 4. Previously, they have also been published in [VaSS02a, VaSS02b, VSG+04, VSGT03]. 1. Introduction 14 Mapping Conceptual to Logical Models. We confront the issues presented in the previous subsection by describing a semi-automatic transition from conceptual to logical model for ETL processes. At first, we determine the appropriate mappings. The constituents of the conceptual model are mapped to their respective constituents of the logical model. Concepts and attributes are mapped to recordsets and attributes. Transformations and constraints are mapped to activities. Notes are used for the determination and instantiation of the appropriate activity template. Also, we tackle two design problems that need special treatment: the case of attributes that are projected-out from the data flow, and the convergence of two separate data flows at a common data store. Then, we provide a method for the semi-automatic determination of a correct execution order of the activities in the logical model, wherever this is feasible, by grouping the transformations of the conceptual design into stages. Finally, we present a methodology for the transition from the conceptual to logical model. Our results are presented in Chapter 5. ETL Workflow Optimization. In order to efficiently execute an ETL workflow and handle the problems mentioned in the previous subsection, we delve into the optimization of ETL processes, in terms of logical transformations of the workflow. To this end, we devise a method based on the specifics of an ETL workflow that can reduce its execution cost by changing either the total number or the execution order of the processes. We set up the theoretical framework for the problem, by modeling it as a state space search problem, with each state representing a particular design of the workflow as a graph. The nodes of the graph represent activities and data stores and the edges capture the flow of data among the nodes. Since the problem is modeled as a state space search problem, we define transitions from one state to another that extend the traditional query optimization techniques. We prove the correctness of the introduced transitions and provide details on how states are generated and the conditions under which transitions are allowed. Finally, we provide algorithms towards the optimization of ETL processes. First, we use an exhaustive algorithm to explore the search space in its entirety and find the optimal ETL workflow. Then we introduce greedy and heuristic search algorithms to reduce the search space that we explore, and demonstrate the efficiency of the approach through a set of experimental results. Our results are presented in Chapter 6. Previously, they have been published in [SiVS05]. Software Construction. To complement the aforementioned issues, we have developed a prototype, named ARKTOS II, with the goal of facilitating the design of ETL workflows, based on the proposed methods and models. 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 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. In Chapter 7, we present the prototype ETL tool ARKTOS II. Previously, a part of ARKTOS II has been published in [VSG+04, VSGT03]. 1.4 Contributions and Outline of the Dissertation 15 In summary, the main contributions of our research are: − The provision of a novel conceptual model which is customized for the tracing of inter-attribute relationships and the respective ETL transformations in the early stages of a data warehouse project along with a methodology for its construction. − The provision of a novel logical model for the representation of ETL processes with two main characteristics: genericity and customization. − The presentation of a methodology for the semi-automatic transition from the conceptual to the logical model for ETL processes. − The presentation of a palette of several templates, representing frequently used ETL processes along with their semantics and their interconnection. In this way, construction of ETL workflows, as a flow of these processes, is facilitated. − The tuning of an ETL workflow through several algorithms for the optimization of the execution order of the activities. − To replenish the aforementioned issues, we have prototypically implemented an ETL tool with the goal of facilitating the design, the (re-)use, and the optimization of ETL workflows. The outline of this dissertation is as follows. In Chapter 2, we present the state of the art and practice for ETL tools. In Chapter 3, we propose a novel conceptual model along with a methodology that facilitates its construction. In Chapter 4, we propose a formal logical metamodel as a logical abstraction of ETL processes. In Chapter 5, we present a semi-automatic transition from conceptual to logical model. In Chapter 6, we delve into the optimization of ETL workflows. In Chapter 7, we present ARKTOS II, a prototype ETL tool. Finally, in Chapter 8, we conclude our thesis with a summary of contributions and future research directions.