

Data Conversion

DATA CONVERSION

This article does not cite any references or sources. (February 2007)

Please help improve this article by adding citations to reliable sources. Unverifiable material may be challenged and removed.

Data conversion is the conversion of one form of computer data to another--the changing of bits from being in one format to a different one, usually for the purpose of application interoperability or of capability of using new features. At the simplest level, data conversion can be exemplified by conversion of a text file from one character encoding to another. More complex conversions are those of office file formats, and conversions of image and audio file formats are an endeavor that is beyond the ken of ordinary computer users.

Information basics

Before any data conversion is carried out, the user or application programmer should keep a few basics of computing and information theory in mind. These include:

1. Information can easily be discarded by the computer, but adding information takes effort.
2. The computer can add information only in a rule-based fashion; most users want additions of information that can only be accomplished by humans.
3. Up sampling the data or converting to a more feature-rich format does not add information; it merely makes room for that addition, which usually a human must do.

For example, a true color image can easily be converted to grayscale, while the opposite conversion is a painstaking process. Converting a Unix text file to a Microsoft (DOS/Windows) text file involves adding information, namely a CR (hexadecimal 0D) byte before each LF (0A) byte, but that addition is easily done with a computer, since it is rule-based; whereas the addition of color information to a grayscale image cannot be done programmatically, since only a human knows which colors are needed for each section of the picture--there are no rules that can be used to automate that process. Converting a 24-bit PNG to a 48-bit one does not add information to it, it only pads existing RGB pixel values with zeroes, so that a pixel with a value of FF C3 56, for example, becomes FF00 C300 5600. The conversion makes it possible to change a pixel to have a value of, for instance, FF80 C340 56A0, but the conversion itself does not do that, only further manipulation of the image can. Converting an image or audio file in a loss format (like JPEG or Vorbis) to a lossless (like PNG or FLAC) or uncompressed (like BMP or WAV) format only wastes space, since the same image with its loss of original information (the artifacts of loss compression) becomes the target. A JPEG image can never be restored to the quality of the original lossless image from which it was made, no matter how much the user tries the "JPEG Artifact Removal" feature of his or her image manipulation program.

Because of these realities of computing and information theory, data conversion is more often than not a complex and error-prone process that requires the help of experts. It is safe to say that only the success of artificial intelligence could put data conversion companies out of a job.

Pivotal conversion

Data conversion can occur directly from one format to another, but many applications that convert between multiple formats use a pivotal encoding by way of which any source format is converted to its target. For example, it is possible to convert Cyrillic text from KOI8-R to Windows-1251 using a lookup table between the two encodings, but the modern approach is to convert the KOI8-R file to Unicode first and from that to Windows-1251. This is a more manageable approach: an application specializing in character encoding conversion would have to keep hundreds of lookup tables, for

Data Conversion

all the permutations of character encoding conversions available, while keeping lookup tables just for each character set to Unicode scales down the number to a few tens.

Pivotal conversion is similarly used in other areas. Office applications, when employed to convert between office file formats, use their internal, default file format as a pivot. For example, a word processor may convert an RTF file to a WordPerfect file by converting the RTF to OpenDocument and then that to WordPerfect format. An image conversion program does not convert a PCX image to PNG directly; instead, when loading the PCX image, it decodes it to a simple bitmap format for internal use in memory, and when commanded to convert to PNG, that memory image is converted to the target format. An audio converter that converts from FLAC to AAC decodes the source file to raw PCM data in memory first, and then performs the lossy AAC compression on that memory image to produce the target file.

Lossy and inexact data conversion

For any conversion to be carried out without loss of information the target format must support the same features and data constructs present in the source file. Conversion of a word processing document to a plain text file necessarily involves loss of information, because plain text format does not support word processing constructs such as marking a word as boldface. For this reason, conversion from one format to another that has fewer features is rarely carried out, though it may be necessary for interoperability, e.g. converting a file from one version of Microsoft Word to an earlier version for the sake of those who do not have the latest version of Word installed.

Loss of information can be mitigated by approximation in the target format. There is no way of converting a character like ä to ASCII, since the ASCII standard lacks it, but the information may be retained by approximating the character as ae. Of course, this is not an optimal solution, and can impact operations like searching and copying; and if a language makes a distinction between ä and ae, then that approximation does involve loss of information.

Data conversion can also suffer from inexactitude, the result of converting between formats that are conceptually different. The WYSIWYG paradigm, extant in word processors and desktop publishing applications, versus the structural-descriptive paradigm, found in SGML, XML and many applications derived there from, like HTML and MathML, is one example. Using a WYSIWYG HTML editor conflates the two paradigms, and the result is HTML files with suboptimal, if not nonstandard, code. In the WYSIWYG paradigm a double line break signifies a new paragraph, as that is the visual cue for such a construct, but a WYSIWYG HTML editor will usually convert such a sequence to

, which is structurally no new paragraph at all. As another example, converting from PDF to an editable word processor format is a tough chore, because PDF records the textual information like engraving on stone, with each character given a fixed position and line breaks hard-coded, whereas word processor formats accommodate text reflow. PDF does not know of a word space character--the space between two letters and the space between two words differ only in quantity. Therefore, a title with ample letter-spacing for effect will usually end up with spaces in the word processor file, for example INTRODUCTION with spacing of 1 em as I N T R O D U C T I O N on the word processor.

Open vs. secret specifications

Successful data conversion requires thorough knowledge of the workings of both source and target formats. In the case where the specification of a format is unknown, reverse engineering will be needed to carry out conversion. Reverse engineering can achieve close approximation of the original specifications, but errors and missing features can still result. The binary format of Microsoft Office documents (DOC, XLS, PPT and the rest) is undocumented, and anyone who seeks interoperability with those formats needs to reverse-engineer them. Such efforts have so far been fairly successful, so that most Microsoft Word files open without any ill-effect in the competing OpenOffice.org Writer, but the few that don't, usually very complex ones, utilizing more obscure features of the DOC file format, serve to show the limits of reverse-engineering.

DATA INTEGRATION

Data Conversion

Data integration is the process of combining data residing at different sources and providing the user with a unified view of these data. This process emerges in a variety of situations both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories). Data integration appears with increasing frequency as the volume and the need to share existing data explodes. It has been the focus of extensive theoretical work and numerous open problems remain to be solved. In management practice, data integration is frequently called Enterprise Information Integration.

History

Figure 1: Simple schematic for a data warehouse. The information from the source databases is extracted, transformed then loaded into the data warehouse.

Figure 1: Simple schematic for a data warehouse. The information from the source databases is extracted, transformed then loaded into the data warehouse.

The problem of combining heterogeneous data sources under a single query interface is not a new one. The rapid adoption of databases after the 1960s naturally led to the need to share or merge existing repositories. This merging can be done at several levels in the database architecture. One popular approach is Data warehousing (see figure 1). Here data from several sources are extracted, transformed, and loaded into source and can be queried with a single schema. This can be perceived architecturally as a tightly coupled approach because the data reside together in a single repository at query time. Problems with tight coupling can arise with the "freshness" of data, for example when an original data source is updated, but the warehouse still contains the older data and the ETL process needs to be executed again. It is also difficult to construct data warehouses when you only have a query interface to the data sources and no access to the full data. This problem frequently arises when integrating several commercial query services like travel or classified advertisement web applications.

Figure 2: Simple schematic for a data integration solution. A system designer constructs a mediated schema over which a user poses queries. The source databases are interfaced with wrapper code if needed.

The recent trend in data integration has been to loosen the coupling between data. Here the idea is to provide a uniform query interface over a mediated schema (see figure 2). This query is then transformed into specialized queries over the original databases. This process can also be called as view based query answering because we can consider each of the data sources to be a view over the (nonexistent) mediated schema. Formally such an approach is called Local As View (LAV) — where "Local" refers to the local sources/databases. An alternate model of integration is one where the mediated schema is designed to be a view over the sources. This approach called Global As View (GAV) — where "Global" refers to the global (mediated) schema — is often used due to the simplicity involved in answering queries issued over the mediated schema. However, the obvious drawback is the need to rewrite the view for mediated schema whenever a new source is to be integrated and/or an existing source changes its schema.

Some of the current work in data integration research concerns the Semantic Integration problem. This problem is not about how to structure the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. For example if two companies merge their databases, certain concepts and definitions in their respective schemas like "earnings" inevitably have different meanings. In one database it may mean profits in dollars (a floating point number), while in the other it might be the number of sales (an integer). A common strategy for the resolution of such problems is the use of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach is also called ontology based data integration.

Example

Consider a web application where a user can query a variety of information about cities such as crime statistics, weather, hotels, demographics, etc. Traditionally, the information must exist in a single database with a single schema. Information

Data Conversion

of this breadth, however, is difficult and expensive for a single enterprise to collect. Even if the resources exist to gather the data, it would likely duplicate data in existing crime databases, weather websites, and census data.

A data integration solution may address this problem by considering these external resources as materialized views over a virtual mediated schema, resulting in "virtual data integration". This means application developers construct a schema to best model the kinds of answers their users want. This virtual schema is called the mediated schema. Next, they design "wrappers" or adapters for each data source, such as the crime database and weather website. These adapters simply transform the local query results (those returned by the respective websites or databases) into an easily processed form for the data integration solution (see figure 2). When an application-user queries the mediated schema, the data integration solution transforms this query into appropriate queries over the respective data sources. Finally, the results of these queries are combined into the answer to the user's query.

A convenience of this solution is that new sources can be added by simply constructing an adapter for them. This contrasts with ETL systems or a single database solution where the entire new dataset must be manually integrated into the system.

Theory of Data Integration

The theory of data integration is a subset of database theory and formalizes the underlying concepts of the problem in first-order logic. Its results tell us whether data integration is possible and how difficult it is to perform. While its definitions may appear abstract, they are general enough to accommodate all manner of integration systems.

Definitions

Data integration systems are formally defined as a triple $\langle G, S, M \rangle$ where G is the global (or mediated) schema, S is the heterogeneous set of source schemas, and M is the mapping that maps queries between the source and the global schemas. Both G and S are expressed in languages over alphabets comprised of symbols for each of their respective relations. The mapping M consists of assertions between queries over G and queries over S . When users pose queries over the data integration system, they pose queries over G and the mapping then asserts connections between the elements in the global schema and the source schemas.

A database over a schema is defined to be a set of sets, one for each relation (in a relational database). The database corresponding to the source schema S would be the set of sets of tuples for each of the heterogeneous data sources and is called the source database. Note that this single source database may actually be a collection of disconnected databases. The database corresponding to the virtual mediated schema G is called the global database. The global database must satisfy the mapping M with respect to the source database. The legality of this mapping depends on the nature of the correspondence between G and S . Two popular ways to model this correspondence are Global as View or GAV and Local as View or LAV.

Figure 3: Illustration of tuple space of the GAV and LAV mappings. In GAV, the system is constrained to the set of tuples mapped by the mediators while the set of tuples expressible over the sources may be much larger and richer. In LAV, the system is constrained to the set of tuples in the sources while the set of tuples expressible over the global schema can be much larger. Therefore LAV systems must often deal with incomplete answers.

Figure 3: Illustration of tuple space of the GAV and LAV mappings. In GAV, the system is constrained to the set of tuples mapped by the mediators while the set of tuples expressible over the sources may be much larger and richer. In LAV, the system is constrained to the set of tuples in the sources while the set of tuples expressible over the global schema can be much larger. Therefore LAV systems must often deal with incomplete answers.

In GAV, the global database is modeled as a set of views over S . In this case M associates to each element of G a query over S . Query processing becomes a straightforward operation because the associations between G and S are well-

Data Conversion

defined. The burden of complexity is placed on implementing mediator code instructing the data integration system exactly how to retrieve elements from the source databases. If any new sources are added to the system, considerable effort may be necessary to update the mediator, thus the GAV approach should be favored in cases where the sources are not likely to change.

In a GAV approach to the example data integration system above, the system designer would first develop mediators for each of the city information sources and then design the global schema around these mediators. For example, consider if one of the sources was for a weather website. The designer would likely then add a corresponding element for weather to the global schema. Then the bulk of effort is to write the proper mediator code that will transform predicates on weather into a query over the weather website. This effort can be complicated if there is another source related to weather because the designer is charged with the task of writing code to properly combine the results from the two sources.

On the other hand, in LAV, the source database is modeled as a set of views over G . In this case M associates to each element of S a query over G . Here the exact associations between G and S are no longer well-defined. As is illustrated in the next section, the burden of determining how to retrieve elements from the sources is placed on the query processor. The benefit of an LAV modeling is that new sources can be added with far less work than in a GAV system, thus the LAV approach should be favored in cases where the mediated schema is not likely to change.

In an LAV approach to the example data integration system above, the system designer designs the global schema first and then simply inputs the schemas of the respective city information sources. Consider again if one of the sources was for a weather website. The designer would add corresponding elements for weather to the global schema only if none existed already. Then an adapter or wrapper for the website would be written and a schema description of the website's results added to the source schemas. The complexity of adding the new source is moved from the designer to the query processor.

Query Processing

The theory of query processing in data integration systems is commonly expressed using conjunctive queries. One can loosely think of a conjunctive query as a logical function applied to the relations of a database such as " $f(A,B)$ where $A < B$ ". If a tuple or set of tuples is substituted into the rule and satisfies it (makes it true), then we consider that tuple as part of the set of answers in the query. While formal languages like Datalog express these queries concisely and without ambiguity, common SQL queries are classified as conjunctive queries as well.

An important property of conjunctive queries (in terms of data integration) is query containment. A query A contains another query B (denoted $A \supseteq B$) if the results of applying B are a subset of the results of applying A for any database. The two queries are said to be equivalent if the resulting sets are equal for any database. This is important because in both GAV and LAV systems, the user's conjunctive queries are posed over a virtual schema represented by a set of views, or "materialized" conjunctive queries. Integration seeks to rewrite the queries represented by the views to make their results equivalent or maximally contained by our user's query. This corresponds to the problem of answering queries using views (AQUV).

In GAV systems, a system designer writes mediator code to define the query rewriting. Each element in the user's query corresponds to a substitution rule just as each element in the global schema corresponds to a query over the source. Query processing is simply expanding the sub goals of the user's query according to the rule specified in the mediator and thus the resulting query is likely to be equivalent. While the majority of the work is done beforehand by the designer, some GAV systems such as Tsimmis involve simplifying the mediator description process.

In LAV systems, queries undergo a more radical process of rewriting. This is because there is no mediator to align the user's query with a simple expansion strategy. The integration system must execute a search over the space of possible queries in order to find the best rewrite. The resulting rewrite may not be an equivalent query but maximally contained,

Data Conversion

and the resulting tuples may be incomplete. The MiniCon algorithm is currently the leading query rewriting algorithm for LAV data integration systems.

In general, the complexity of query rewriting is NP-complete. If the space of rewrites is relatively small this is not a problem even for integration systems with hundreds of sources.

Enterprise Information Integration

Enterprise Information Integration (EII) is the commercial application of Data Integration. Unlike the theoretical problems described above, the private sector is more concerned with the problems of data integration as a viable product. Emphasis is neither on correctness nor tractability but on speed and simplicity. The industry for EII has emerged, but many professionals believe it is not performing to its full potential. Practitioners cite the following major problems with EII must be addressed for the industry to become mature:

EII must be simple to understand

Answering queries with views is interesting from a theoretical standpoint, but it is difficult to understand how to incorporate it as an enterprise solution. Some developers believe it should be merged with EAI. Others believe it should be incorporated with ETL systems, citing customers' confusion over the differences between the two services. EII must be simple to employ

Even if EII is recognized as a solution to a problem, it is currently time-consuming and complex to apply most EII software solutions to the problem. A variety of schema-less solution such as "Lean Middleware" has been posed, however ease of use and speed of employment are inversely proportional to the generality of such systems. Others cite the need for standard data interfaces to speed and simplify the integration process in practice. EII must handle higher order information

It is difficult even with a functioning information integration system to determine if a given application will be satisfied by the sources in the database. Answering these kinds of questions about a set of repositories requires semantic information like metadata and/or ontologies. The few commercial tools that leverage this information are in their infancy.