A Methodology for Data Cleansing and Conversion

Leslie M. Tierstein,
W R Systems, Ltd.

Overview

This document presents a methodology for transferring data from one or more legacy systems into newly deployed application databases or data warehouses. The methodology incorporates two interrelated and overlapping tasks:

?? "cleansing" the legacy data so that it can be
?? "converted" into a production database in use for a new, target application or data warehousing effort.

The methodology can be applied to any data cleansing and conversion effort, but includes special consideration for transferring data to Oracle databases – both databases deployed as part of custom development efforts and those installed as part of an Oracle Applications suite.

Terminology

This paper uses the following terms:

Term	Definition
legacy database	Database used by an existing, legacy system, which must be cleansed and converted for use by a new application or data warehouse
legacy system	Computerized system currently in place whose data must be cleansed and converted for use in a new application or data warehouse
target system	Application or data warehouse which will be installed and will use the newly constructed database
target database	Unified database containing cleansed data from the legacy database(s) and used by the target system
development database	Target database instance used during system development and initial testing
production database	Target database instance into which cleansed and converted legacy data will be converted for deployment of the target system

Primary Goal of Data Cleansing

The primary goal of a data cleansing effort is to eliminate data inconsistencies, invalid values, and other shortcomings in data integrity from the legacy databases. This will greatly facilitate converting the existing data to the format required for the target system.

The objective is to clean the data "in place", that is, to correct the data stored in the legacy databases. If this objective can be met:

?? the cleansed data will be available for use as long as the legacy system is operational.
?? the cleansed data will be available for archiving when legacy systems are retired.
?? the cleansed data can ported to the target system.

If this objective cannot be met (for reasons to be discussed later in this document), the goal is to identify the defects and find another way to eliminate them. Corrections will be applied after the data has been ported to the target system, but before it is inserted into the production database.

Secondary Goals

Personnel participating in a data cleansing project need to know about both legacy and target data structures. Taking part in such a project allows legacy system experts to familiarize themselves with the target database, some of the development tools, and some target system functionality. It allows target system development personnel to familiarize themselves with the contents and structure of the legacy database, and to verify that the target system supports required data and relationships.

The data cleansing and conversion effort cannot be seen as the stepchild of the development project. It's often the highest risk area of the entire deployment effort. If conversion is granted its proper importance (and staffing), a powerful synergy can develop between development and conversion personnel –something a conversion analyst learns can directly help the development team, and vice versa.

Data Cleansing Methodology

This data cleansing methodology is based on using a "targeted search" strategy to detect and correct invalid or incomplete data. The target is the new database structure. "Dirty" data can be found by

tracing back from the new structure to the source(s) of the data.

This approach is the most effective means to detect "dirty" data in a large-volume database to be converted to a new format. It not only targets data most critical to the development effort; it also allows work to be done in the development environment, rather than on the operational system(s) where the legacy applications reside.

Components

The data cleansing effort consists of the following components:

- ?? Planning and preparation
- ?? Legacy-to-target database mapping
- ?? Designing, testing, and executing programs to detect data which needs to be cleansed
- ?? Analyzing the results of the data cleansing run(s)
- ?? Legacy data cleansing, including identification of manual cleansing needs and strategies for providing missing or incomplete data.

Planning and Preparation

The planning and preparation phase of the data cleansing effort consists of the following tasks:

- ?? Prepare a schedule and work plan.
- ?? Specify hardware and software needed.
- ?? Prepare a Configuration Management Plan.
- ?? Specify project metrics.

Schedule and Work Plan

You can't realistically start by preparing a detailed schedule – there are too many unknowns. Since this methodology uses a targeted search strategy, the amount of time required to cleanse data is directly proportional to the number and complexity of the mappings and transforms which must be applied to the source data.

An especially tricky part of doing any planning is that the target database is usually a "moving target" – it is being developed at the same time as the cleansing and conversion software, and is always subject to change. If the new software is not a custom application, but a commercially available package, such as the Oracle applications, you're a little luckier – but flexfields and extensions are still subject to change.

So, start with a project plan and pert chart, showing tasks involved, dependencies between the tasks, and the staff required to peform each task. In addition to development functional and technical experts, you'll need help from legacy system experts. Take the time to familiarize legacy systems' operations and MIS staff with the data cleansing effort, and the tasks required from them. As part of the planning for cleansing each subsystem's data, schedule time for tasks for which legacy system staff participation will be required.

Hardware and Software Specifications

Specify, install, and configure the hardware and software required to carry out the data cleansing and conversion efforts on both the legacy system(s) and the development machine. For example, telecommunications links, ftp software, and/or compatible media may be required to transfer data from a source to target machine. Software may be required to translate data from EBCDIC to ASCII. A new database instance, to contain cleansed and/or converted data, needs to be sized and created.

More critical, you should have an application to directly support the cleansing and conversion effort. This software should be able not only to document the mappings, but also to generate code to implement the cleansing and conversion programs.

Several COTS (commercial off-the-shelf) packages are available to do this job. However, these seem to be quite expensive, and tailored to mainframe environments. A good alternative is to consult the Proceedings of the Oracle Development Tools Users Group (ODTUG) conferences: several participants have suggested solutions using Designer/2000 and other Oracle-supplied development tools.

Configuration Management Plan

The Configuration Management (CM) Plan typically specifies how versions of the software will be managed. Be sure the plan includes a detailed procedure for controlling software changes. For example, a Change Control Board (CCB) may be formed to review proposed database changes. This is critical to ensure communication between the software development team and the cleansing/conversion team.

Metrics

Specify the criteria to be used to measure the data quality and for determining whether the data is valid or needs to be cleansed.

Legacy-Target Mapping

Using a targeted search strategy, analysts familiar with the legacy database structure and functionality work together with those developing the target database to determine precisely how each data element is mapped to its new destination(s).

Table-Level Mapping

The first task in the legacy-target mapping is to review each table used by each legacy application and to determine if the data in the table is needed in the new application. (In this document, the term "table" is used to indicate a data store, whether it is implemented as a file, database table, or other storage format.)

It is important to explicitly specify and report on the source data which is *not* needed in the target system, and to note the reason why. This allows legacy personnel to more effectively review the proposed mappings, and gives a level of confidence that data hasn't been overlooked. For example, a table used as a temporary batch transfer mechanism between two applications probably contains data which duplicates other table contents, and does not need to be cleansed or converted.

The proposed table-level mappings may reveal one-to-many or many-to-one mappings.

In a one-to-many mapping, one legacy table yields data to populate multiple target tables. This is typically the case when a database which is not in third normal form is converted to normalized structure.

In a many-to-one mapping, multiple source tables contribute data to the same target table. For example, when several "stovepipe" systems are converted, some data may be duplicate and can be discarded. Further analysis is always required for these mappings. Legacy functional and database experts must determine which set of data is to be maintained. Alternatively, analysis may reveal that some elements exist in only one source application, and need to be added to data retrieved from other source(s) to yield all data necessary for the target application.

Target Tables

Oracle applications provide "open interfaces", which offer a semi-automated means to populate portions of the application database. Open interfaces allow users to import detailed transaction data from external systems. The data is loaded into specially designated interface tables. These tables are processed by concurrent interface programs, which apply the same validation criteria as the standard application modules, and insert clean data into the application tables. Records which fail to meet the validation criteria are marked as rejected and can be manually corrected and reprocessed.

When no interface tables are available (which will be true most of the time), legacy data must be mapped directly to the tables in the target database. Even when interface tables are provided, it is instructive to initially map the legacy tables to the application tables where the data will end up. This will make the cleansing requirements more clear to legacy users and developers alike.

Column-Level Mapping

The next task in the legacy-target mapping is to review each proposed table-level mapping and to specify the column mappings between the two tables. During this stage, inconsistencies between legacy and target column formats may be detected. For example, the lengths of character columns may not match; or the target column includes domain ("value set"), range, or other validation not applied to the legacy data. The nature of the discrepancy, if any, and the data "transform" which must be applied to correct the discrepancy must be noted.

Column-Level Transforms

One of the most difficult cleansing and conversion tasks is resolving discrepancies between the source data value and the format and/or semantics specified for its target column. Such discrepancies are typically resolved by transforming the legacy data into a new value.

Some transforms may be fairly straightforward. For example, a set of 2-letter codes could be changed to an equivalent set of 5-letter codes via a simple formula (DECODE…) or lookup embedded in a conversion program.

The probability for complications is greatly increased when tables from multiple legacy systems are merged into one target database and/or the format or structure of primary and foreign key fields must be changed. Examples of some of these complications are given in the following sections.

Consider a legacy system which includes vendors and contracts, where each contract must track the responsible vendor. If the format of the vendor number (the primary key for the vendor table) is

invalid for the new system, it must be transformed to a new value. (For example, it may include letters, and the target system supports numeric-only vendor numbers.) Transforming this value only in the vendor table is insufficient; the foreign key constraint to the vendor number in the contract table must also be transformed. To do so, a cross-reference between the original value and the newly assigned value must be maintained.

As another example, consider two applications which use a code for the account type. However, the same code has different meanings in the two applications. The cleansing/conversion software must be aware of the source of the data, and transform the incoming code from only one of the source applications to a new value.

The supporting software must be able to document the required transform rules and, eventually, to generate code to implement these rules.

Finding Defective or Missing Data

Cleansing the data for each application consists of the following phases:

- ?? Analysis and conceptual design
- ?? Detailed design and coding
- ?? Inspecting legacy data to detect defects
- ?? Analysis of results of the data inspection
- ?? Further data cleansing, if required
- ?? Reports on the cleansing effort.

Analysis and Conceptual Design

The analysis and high-level design phase entails the tasks listed below.

- ?? Develop standards.
- ?? Develop a list of tables to be cleansed.
- ?? Develop a list of missing tables.
- ?? Develop a sampling strategy.
- ?? Get the team together.

Standards. Develop or adopt standards for writing the cleansing and conversion software. Standards documents must include naming guidelines, coding standards for all languages to be used, and procedures for writing, reviewing, and accepting software.

List of tables to be cleansed. Based on the legacy-target mappings specified earlier, develop a list of the legacy tables to be cleansed. The tables should be listed in the order in which they will be cleansed (and subsequently converted). This will also be the order in which the legacy data will be delivered to the developers. At this point, the order is preliminary only. However, it will form the basis for the conversion order which is critical. For example, in cases where referential integrity must be established, the parent side of a relationship must be available in order to create the child records. In cases when two or more tables hold similar or redundant information, determine which table will be the primary source of data and which, will provide additional records or column values which will update previously created records.

List of missing data. Based on the target-legacy mappings specified earlier, determine which data, if any, appears to be missing from the legacy systems. At this time, "missing data" will be any target table which does not have a legacy source, This might not be a real problem, but now is the time to find this out, by figuring out why the data appears to be missing. For example

Type of Missing Data	Resolution
Represents new functionality	Add data entry to the list of startup tasks for the users
Relates to system configuration (for example, printers)	Add to the list of configuration tasks for developers
Really missing!	Reexamine the list of legacy tables to be sure you haven't missed anything; if you haven't, start planning for getting the data

Sampling Strategy. Develop a sampling strategy, to determine an appropriate subset of data to be examined in the data cleansing effort. Given time and resource constraints, it will probably not be possible to examine every record in the legacy tables. However, by extracting a representative sampling of records, the data cleansing task should be able to detect trends. The data could be selected by age -- for example, a range of current vs. completed transactions could be examined, or records of different types may be included. The sampling algorithms must ensure that sets of related records are selected (when parent-child relationships are involved), and that a representative

sample of record types (for example, different types of inventory items or transactions) is selected.

Teamwork. Part of the high risk of data cleansing is the high degree of teamwork required between legacy experts, the conversion team, development personnel, and support personnel.

- ?? Coordinate with the development team and database administrator to ensure that the database instance and database structures will be in place to receive the data to be cleansed and, later, converted.
- ?? Have conversion team and legacy system experts formally review the format of the data to be extracted, so that the software can be produced to match this format.
- ?? Make sure everyone (development teams and legacy system staff) has bought into the configuration management plan and procedures. People from all organizations must be in place to monitor changes to the target database structure and the structure of the legacy data extracts, in order to assess the impace of these changes on the cleansing and conversion software.

Design and Code

During the design and code phase, the conversion team has to develop, test, and document software and/or procedures to:

- ?? Extract data from the legacy tables according to the specified sampling strategy
- ?? Transfer data from the legacy system(s) to the system where the cleansing effort will be conducted.
- ?? Load the data into the conversion or open interface tables
- ?? Find and report on any dirty data or potential problems.

The methodology for developing these programs must incorporate an approach that will work, efficiently, all the time, with minimal high-tech muss-and-fuss. The chosen methods must be able to convert the legacy data from whatever format it's in – proprietary or non-proprietary; hardware- and operating system-dependent or independent; database or flat file; hierarchical, networked, or relational; state-of-the-art or obsolete; ASCII or EBCDIC – into ASCII, human-readable text files. These files are then input to the cleansing and conversion programs on the target system. Write software which is prepared for the worst – assume you will have to convert data from a database system that is proprietary, obsolete, badly documented, machine-dependent, has no ODBC or other external drivers, and has no published API.

Extraction Programs. Extraction programs must dump the data in the legacy tables to be converted to a form which can be loaded into the target database. In an Oracle environment, the most straightforward approach is to produce ASCII fixed-format records which are input to SQL*Loader. The extraction effort is complicated by the fact that Oracle has a hard time deciphering many machine-dependent datatypes found on mainframes or other proprietary hardware. Such data must be translated to human-readable (EXTERNAL) datatypes. Some legacy systems may include utilities to do this, but others might need to have custom programs written.

Transfer Programs. Transfer programs copy the legacy data to the target system where subsequent cleansing will be performed. Transfer programs can be as simple as ftp'ing a file from one networked computer to another. In an environment including mainframes or networks which are not Ethernet-based, however, they may get more compex, and require the purchase and configuration of third-party software.

Load Programs. Load programs will be SQL*Loader scripts, which use SQL*Loader control files to load data from a flat, ASCII file into the specified database table(s). The SQL*Loader scripts use the options to create a BAD file, to contain all records which did not successfully load, and a LOG file, to log all rows processed. The tables into which SQL*Loader inserts the data are not part of the target database; rather, they are temporary tables whose purpose is to hold all data from the legacy systems in a format accessible by the Oracle development and query tools.

Reports. Write programs to report on any records which were loaded into the temporary tables, but have defects which would not be acceptable to the production database. One program should be written for each table-level mapping. The analysis should include:

- ?? the source of the problem (the table and column from which the defective data was extracted)
- ?? the extent of the problem (the percentage of records in which the problem was detected)

?? the nature of the defect, and the record in which it occurred.

Potential defects include but are not limited to:

?? absence of a value in a required field
?? violation of a range or check constraint
?? violation of a formatting rule
?? invalid value for a column which has value set (domain) validation.

Producing these reports should be at least partially automated by the support software. For example, if the development effort is using Designer/2000, domains, ranges, check constraints,and formatting rules are associated with the column to which they apply in the repository. Scripts can be written to check that any legacy column mapped to that target column meets the specified validation criteria. Similarly, the Oracle applications repository includes information on the value sets and formatting rules to be applied to database columns. This information is stored in the FND tables, and scripts can be generated based on the specified rules and values.

Analyzing Results

The analysis phase will entail the tasks listed below.

1. If required, finish designing, coding, and testing the analysis reports.
2. Run the reports. Print and distribute the resulting analyses.
3. Determine the cause of the defective data and the extent of the problem.
4. Based on the nature of the dirty data detected, determine the strategy to be used to clean up the defects. Four cleansing strategies are available, depending on where and how the cleansing is done:
   ?? Automatic data cleansing on the legacy system
   ?? Manual cleansing on the legacy system
   ?? Automatic data cleansing on the target system, as part of loading and converting the data
   ?? Manual cleansing on the target system.

Means must also be provided to supply any missing data detected by the analysis.

Cleansing Data on the Legacy System

Cleansing the data in place in the legacy systems would improve the quality of the legacy system data. It would also allow data cleansing to be performed prior to and independent of the conversion effort.

Cleansing Legacy Data "Automatically"

The simplest fix is to write a program to cleanse the bad data. The program, which can be run while normal operations continue, would change invalid values to valid values. For this strategy to succeed, the records with faulty data must be identifiable based solely on information stored in the row containing the defective data or in related rows in other tables.

Before such a program is written or run, a legacy system expert needs to identify the source of all defective data:

?? All records in which the defective data occurs (not just those examined in the sample).
?? The program or programs which caused the defects to be introduced.

The dirty data may have resulted from previous unvalidated data entry and/or programming errors which have since been corrected. In this case, replacing defective values with clean data should entail no changes to legacy application software, and a cleansing program can be written and run. (If the system has been in operation forever, an aging analysis might be included in the analysis programs. Such an analysis might reveal, for example, that data created before 1978 [really!] is hopeless incomplete. Users might be forced to conclude that theyr eally don't need 20 years of history converted, thereby changing slightly the scope of the conversion effort.)

If the dirty data resulted from defective software that is still in operation, the software should be identified, and an estimate made for fixing the software and cleansing the data. However, if extensive software changes to legacy systems would be required in order operate using the cleansed data, it might not be practical (given time and resource constraints) to correct the data on the legacy systems. In this case, the cleansing code would have to be executed as part of the data conversion.

Cleansing Legacy Data "Manually"

In some cases, the process of correcting the faulty data cannot be automated. For example, if a field contains an invalid date or is missing data which is required by the new system, and the record or related records offer no clues as to what the correct data should be, it is not possible to write generalized code to correct the problem. Manual intervention would be required. Analysts would have to refer to written documents or other records to determine the correct data. Data could then be entered via programs which update an individual record with the appropriate value(s). This approach is, obviously, suited primarily to low-volume application and/or cases when the accuracy of the missing data is critical to functionality.

Cleansing Legacy Data in the Target Environment

Data can be cleansed on the development system if this is the most time- and cost-effective way to proceed. Pending the analysis of the causes and extent of the defective data, the scripts which convert data from the conversion tables to the target database can be written to correct the data and/or to supply missing values. SQL*Loader scripts can also incorporate simple transforms.

Due the nature and volume of most cleansing and conversion efforts, it is impractical to include individual record updates to data that cannot automatically be fixed. In this case, columns can be supplied with default values, so that they do not violate referential or other integrity constraints placed on the target database.

Supplying Missing Column Values

A legacy column value is classified as "missing" if:

- ?? A target-legacy column mapping report reveals that no source column exists for a particular legacy column. In this case, all legacy data would be missing the value.
- ?? The data cleansing analysis reports reveal that some legacy records have null or blank values for a column which is required (NOT NULL) in the target table.

Various approaches may be taken to supply missing data. The legacy staff and conversion team need to determine which approach best suits each instance, and to make additional design and implementation decisions, as outlined below.

- ?? The conversion software automatically supplies a valid default value.
- ?? The conversion software automatically supplies an "invalid value" code.
- ?? Personnel manually enter additional data which the conversion software must convert.

Deriving Valid Default Values

The conversion software can automatically supply a valid default value. This approach is best used if the column represents new functionality and/or no legacy column can be mapped to the target column. In this approach, the column is supplied a valid value (as agreed upon by legacy and development personnel) and is, therefore, ready for online processing. However, analysts reviewing the converted data cannot differentiate between records which originally contained the default value, if any, and those whose values were altered.

This approach can be implemented by supplying an appropriate transform formula to the legacy-target column-level mapping. The formula would be a simple SQL DECODE (or NVL) statement to translate NULL legacy values into the agreed-upon target value.

Using an "Invalid Value" Code

The conversion software can automatically supply a value which is the agreed-upon way to denote an invalid or missing value, but which will be temporarily allowed in the database. For example, a missing transaction code can be replaced by a series of question marks equal in length to the field size. (The same value must also be added to the corresponding value set.) Post-conversion reviewers can then easily find the discrepancies and correct them, provided the application includes software to edit the affected field(s).

This approach can be implemented by supplying an appropriate "transform" formula to the legacy-target column-level mapping. The formula would translate NULL and/or invalid legacy values into the agreed-upon target value. If the set of invalid values is not known, writing the formula may involve programming more complex and time-consuming than that required for supplying default values, as discussed above.

Manually Supplying Missing Values

In some cases, the conversion software cannot automatically supply missing values. This would be the case, for example, if the legacy system stored

procurements only at a summary level and line item details were required. In such a scenario, a temporary table would be established, and a data entry screen provided. All missing data would be entered; since the volume of data is likely to be great, this should be done well before the scheduled cutover. As part of the conversion, this additional data would be inserted into the production database in the appropriate sequence.

The data entry should be done on the target system. This allows legacy personnel to familiarize themselves with the new system, and creates the data in a format requiring the least amount of work to prepare it for use in the new system.

This approach involves the following steps:

1. **Analysis and Design.** Legacy and development personnel agree that values are missing and must be provided manually.
2. **Development.** Development personnel develop data definitions and data entry screens to allow entry of the missing values.
3. **Deployment.** Development personnel install the software on machines accessible to legacy personnel, and teach these personnel how to use the screens.
4. **Data Entry.** Legacy personnel enter the missing information, including the unique identifier(s) which relate the new information to the corresponding row already in the legacy database.
5. **Conversion.** Development personnel convert the additional data to its target format as part of the data conversion effort.

Supplying Missing Rows

A row is classified as "missing" if it would be the parent record of a row in another table which has a foreign key constraint to the missing row. For example, every project must have a project type. If the corresponding project type is not in the database at the time of the conversion, the project record cannot be converted.

Two approaches are available for supplying such missing data:

- ?? Automatic creation of missing data based on existing data. (For Oracle application flexfields, this corresponds to the capability to "Allow dynamic inserts".)
- ?? Manual entry of missing data.

Automatically extracting unique data is easy to program but fraught with difficulties. For example, a simple INSERT/SELECT could extract project codes from a project table:

```
INSERT INTO project_desc
   (proj_code, proj_descr)
(SELECT DISTINCT
     (projcode, projcode)
   FROM legacy_projects);
```

This approach has the following difficulties:

- ?? There is no (automatic) way to supply a (meaningful) description.
- ?? There is no (automatic) way to verify that each project code is, in fact, valid in the new system.

Manually inserting new rows in the target table(s) has the obvious drawback that it may be time- and labor-intensive. However, it may also be more accurate.

Fixing Non-Normalized Data

What if a value is, paradoxically, not only missing, but redundant, with no automated way to select between two or more variants? For example, a non-normalized design doesn't include a company table (missing data); instead, it allowed users to enter a customer (company) name for each invoice, rather than having the invoice refer to a customer. So, the legacy data contains four invoices made out to the customer "ABC Tool"; three to the "ABC Tool Company"; and one each to "ABC Tool Company, Inc.", and "ABC Tool Co., Inc". All these invoices, of course, refer to the same company (redundant data). Trying to do a "SELECT DISTINCT" on these company names would yield four customers, where the target table should contain only one. (The same situation can arise when two normalized systems must be converted to one target application, and each contains the company name in a different form.)

To resolve this problem, you need to set up a cross-reference between one or more legacy values and the new value to be used in the target system. For the example above, the solution might look like:

1. Create a table definition that looks like:

```
CREATE TABLE cust_xref (
    old_cust_name CHAR(40)
  , new_cust_name CHAR(40)
)
```

with the primary key consisting of both the old and new customer names.

2. Seed the cust_xref table with each distinct customer name from the legacy system(s):

```
INSERT INTO cust_xref
   (old_cust_name,
    new_cust_name)
(SELECT DISTINCT
       customer, customer
     FROM legacy_customer)
```

3. Write a data entry screen so the users can change the new customer name so that all the old customers that really refer to the same commercial entity have the same new customer name.

Old Name	New Name
ABC Tool	ABC Tool Co.
ABC Tool Company	ABC Tool Co.
ABC Tool Co., Inc.	ABC Tool Co.

4. Be prepared to write conversion software that creates one customer record for each distinct new_cust_name, assigns a unique identifier to each customer, and uses this identifier throughout the system.

Conversion Methodology

The conversion effort must populate the target database with clean data, that meets all validation and referential integrity constraints imposed by the software and/or the application database.

Components

The conversion effort consists of the following components:

- ?? Analysis and high-level design
- ?? Detailed design and coding
- ?? Testing
- ?? Deployment (the execution of conversion code to insert cleansed data into the production database)
- ?? Post-conversion reports.

Analysis and High-Level Design

The most critical task of the analysis phase of the conversion effort is writing the detailed Conversion Plan.

Order of Conversion

Establishing the order in which tables are to be converted is critical. The order is based on the dependencies between the tables. Typically, reference tables, containing stable information, must be loaded first; then, tables which establish primary keys; then tables when have foreign key constraints to the base tables.

The data cleansing effort and plan laid some groundwork for establishing the order in which the target database must be populated. However, for purposes of full-scale data conversion, inter-application dependencies must be examined in more detail.

Reference Tables

Reference tables are those tables which contain reference data which infrequently changes. Data in such tables is referenced in transaction-based tables, but must be established ahead of time. At the physical level, reference "tables" may be implemented in any number of ways, as Designer/2000 domains, Oracle application value sets, or independent tables.

Even once the order of conversion is established, the mechanism for populating each of the reference tables must be established. This may be considered in the Conversion Plan or the Deployment Plan, or may have been done previously as part of prototyping. Previously specified values may have to be changed or augmented as data cleansing results are collected. Users may, for example, discover a code value which was valid years ago, since declared inactive, but still is used in some of the historical records they need converted.

Data Migration

The Conversion Plan may also have to account for the potential recovery and/or examination of legacy data that does not need to be converted. Such tables may be migrated to the target database with their current structure intact, to allow continued accessibility and query via the Oracle tools.

Detailed Design and Coding

In the detailed design and code phase, the software for each table-mapping must be designed, implemented, unit tested, and documented. Some of the software may have previously been developed and tested as part of the cleansing phase; for example, SQL*Loader scripts and control files will

be reused. However, because the volume of data to be converted is much greater than the data sampled in the cleansing effort, further optimization of the code and database will be required. Further, the conversion software must incorporate all those transforms that were part of the cleansing effort.

Direct Path vs. Record Load

By default, SQL*Loader loads records into the database by doing an INSERT for each record to be loaded. SQL*Loader also includes a DIRECT PATH option, which is much faster than the default. This option writes multiple records to the table's data file, without doing individual SQL INSERT operations. Any indexes are created after all the data has been loaded. The Direct Path option can only be used if:

- ?? Data in the input (ASCII) file is in order by the primary key of the target table.
- ?? The target table has no associated database triggers, or the database triggers can be temporarily disabled and the trigger activity performed by a subsequent program.

SQL vs. PL/SQL

For tables which do not have an Oracle applications open interface, SQL or PL/SQL scripts must be written to insert data from the conversion tables into the production database. These scripts must prevent the attempted conversion of any data which it was previously agreed would be cleansed on the legacy system. The scripts will incorporate any transforms which, it was agreed, could not or would not be performed on the legacy systems.

Processing large volumes of data via SQL INSERT/SELECT commands is faster, by an order of magnitude, than processing the same volumes of data via PL/SQL commands which must use cursors and perform individual INSERT commands. For this reason, SQL should be the implementation language of choice unless a conversion requirement which is not supported by SQL needs to be implemented.

Reports

The design and coding phase must also include the implementation and testing of the post-conversion reports. Such reports can typically be written using SQL*Plus and SQL commands. However, the Developer/2000 Reports component may be required.

Conversion User

Every Oracle application table include "Row Who" columns, which track the date the row was created or last changed, as well as the id of the user who performed the change. If row creation and change data is stored in the legacy table, it will be carried forward into the target table. If the data is not currently maintained, the id of the user who ran the conversion script, and the date of the conversion will be used as the creation date and id. One or more users must be created for the purpose of running conversion scripts

Cross-References

Sometimes, the format of a primary key field must be changed to conform to application standards. (This is discussed in the section of this paper on column-level transforms.) A cross-reference between the old and new values must be maintained during the conversion process, to enable records which have foreign key dependencies (via the primary key value) to use the new primary key of the converted record. If deemed appropriate by the users, the original primary key value can be maintained online after the system goes into production, for example, as part of a comment.

Testing

Testing will incorporate the following phases:

- ?? Unit Test
- ?? Integration Test
- ?? System test or "dry run"

Unit Test

All developers are expected to unit test programs they have written before the programs are made "public". The unit tests will be performed on small samples of data. Unit tests can be repeated, if required, with new samples of data which has been cleansed in accordance with the findings of previous cleansing reports.

Integration Test

After all programs have been unit tested, a new, internally consistent sample of cleansed data should be produced. The programs will then be run, in the order prescribed by the Conversion Plan. Any program errors will be noted and fixed as appropriate. Additional data cleansing problems may also be noted.

System Test or Dry Run

The dry run is the test of the conversion software on a full-database extract. It is critical to the success of the conversion and deployment effort. It provides statistics to be used in planning and scheduling the deployment and in allocating resources to do so. It can also be used to verify that database sizing is correct, and to allow the database administrator and/or application data administrator to tune database or table parameters as required before the production database is configured for online operation.

The dry run entails the tasks listed below.

1. **Revise the extract programs** to perform a full-database extract from the legacy databases. For both the dry run and system deployment, the extract files must include all data to be carried forward to the target system.
2. **Prepare the production database instance** to accept the data load. Since the dry run involves a full-database extract, the production database may be the only available database which is large enough to hold all the data. Develop, test, and document procedures for returning the database to its initial state in order to perform live deployment (or another test).
3. **Extract the cleansed legacy data**. During the extraction process, all source databases must be frozen to ensure that an internally consistent snapshot is obtained.
4. **Transfer the extracted data** to the development machine. Record the personnel, computer resources and elapsed time required to perform the extraction and transfer, and the effect, if any, on performance on the legacy and target systems. This information will be used to schedule the system deployment.
5. **Load the data** into the conversion or interface tables using the SQL*Loader programs previously written and tested. Record the personnel, computer resources, and elapsed time required to perform the load and the effect, if any, on performance on the production system. This information will be used to schedule deployment activities. For example, if loads degrade online performance, testing and other activities may have to be curtailed during this time, or the load transferred to off-peak hours.
6. **Run the conversion programs** in the specified order. These include both custom programs and Oracle applications feeder programs. Record the personnel, computer resources, and elapsed time required to perform the conversions and the effect, if any, on target system performance. This information will be used to schedule deployment activities and to pinpoint processes whose performance must be improved via database or application tuning.
7. **Examine the results of the conversion**, noting any software fixes that need to be made to the extract, load, or conversion software. To rigourously review conversion results, test the new applications using the converted data.
8. **Write up analyses of the dry run**.
   - ?? A performance report is required, stating the time required to perform each task in the dry run, broken down for each legacy file and/or each conversion script.
   - ?? A cleansing report is required, produced by running the cleansing reports previously developed, to report on any dirty which is still too dirty to be successfully converted.
9. **Determine how any newly detected dirty data will be fixed** – either on the legacy systems, or additional conversion code. Assign personnel to fix the legacy data, and schedule these fixes.

Post-Conversion Reports

Several types of reports must be provided to adequately report on possible cleansing and conversion failures.

- ?? Log files will be produced by SQL*Loader. Such reports specify the number of records loaded by SQL*Loader, and the number, if any, rejected, and the reason.
- ?? Post-conversion reports list all records which were loaded into the temporary tables, but which failed to be converted into their target format.

SQL*Loader Logs

A detailed example of the SQL*Loader log file format is given in the Oracle 7 Server Utilities Manual, in the section on the "SQL*Loader Log File Reference". The log file includes:

- ?? The name of each column loaded, its character type, and length, as well as its position in the input file.
- ?? A list of any records discarded, and the reason that the record was not able to be

loaded. Reasons for rejection include, but are not limited to:
    - ?? unique constraint violation
    - ?? data type mismatch.
- ?? Summary information on the table load, including:
    - ?? number of rows loaded
    - ?? number of rows rejected due to data errors
- ?? Summary statistics on the load session, including elapsed run time and CPU time.

Conversion Reports

Custom reports will be run after each custom conversion program. The reports will include:

- ?? The legacy-target table mapping which was implemented by the conversion program.
- ?? A list of records which did not convert because of unique constraint violations
- ?? A list of records which did not convert because of referential integrity violations, and the nature of the violation (that is, which foreign key constraint was violated)
- ?? A list of records which did not convert, or for which default values were used, because of logical integrity constraints (that is, constraints which are not enforced at the database level, but are implemented via Designer/2000 domains or Oracle applications value sets), and the nature of the violation.

Oracle-supplied interface tables and programs have corresponding Interface Transfer Reports. Such reports list all records which failed to load, and enumerate the reasons which caused the load to fail.

Conclusion

About the Author

4. Supporting Software

Overview

This document outlines the procedures and programs required for running the software which comprises the Data Cleansing and Conversion Support application.

Assumptions

Using this software requires that detailed table and column definitions for both the legacy files and the target tables reside in the Designer/2000 repository. Further, it requires that users have access to the Designer/2000 software, in order to view, maintain, and/or report on these table and column definitions. Users must also be able to run the application, which uses the Oracle tools SQL*Plus and Developer/2000.

Prospective Users and System Architecture

The application consists of components designed and developed for two different, but potentially overlapping, sets of users:

?? Conversion developers
?? Legacy and functional system experts.

All users, but especially the legacy and functional system experts, will be able to run a set of provided forms and reports to enter, maintain, or report on:

?? Legacy-to-target table-level and column-level mappings
?? Analysis of data cleansing based on the currently specified mappings
?? Logs and other reports produced after a conversion script is run.

Conversion system developers will have access to other components of the support application. These components are, essentially, programs which generate programs. For example, based on column-level legacy-to-target mappings and the validations defined for the target columns, a support application will generate a cleansing script for each legacy-to-target table mapping. The script will check each field in each record in the legacy extract file against the validation rules specified both in the Designer/2000 repository and the transforms specified in the column-level mappings. The generated script is then run, to yield the detailed cleansing analysis.

Continued on next page

4.1 Data Cleansing and Conversion Tasks

Overview

The data cleansing and conversion effort will comprise the tasks summarized below, and explained in detail in the following sections. The tasks are listed in roughly chronological order. However, it is anticipated (and planned) that some tasks should overlap, and that some tasks may be ongoing, as additional information is gathered.

Step	Action
1	Review legacy-target table level mappings, to ensure their accuracy and completeness. ?? Use the **Legacy Tables Report** to list, by application, the legacy tables which are currently specified to be converted. ?? Use the **Legacy Table Mapping Report** to list mappings previously entered, sorted by legacy application and table. ?? Use the **Target Table Mapping Report** to list mappings by target application and table. Enter any additions, deletions, or corrections using the **Legacy-Target Table Mapping Screen**.
2	Review legacy-target column level mappings, to ensure their accuracy and completeness. ?? Use the **Legacy Column Mapping Report** to list mappings previously entered, sorted by legacy application, table, and column. ?? Use the **Target Column Mapping Report** to list mappings by target application, table, and column. Enter any additions, deletions, or corrections using the **Legacy-Target Column Mapping Screen**. Enter any additional target validation rules using the standard Designer/2000 **Edit Table** dialogs or property sheets.
3	Based on the legacy table definitions in the Designer/2000 repository, generate SQL DDL scripts which will define the tables which will hold data loaded from legacy extract files in an Oracle database. The ability to generate DDL is provided by Designer/2000.
4	Based on the legacy table definitions in the Designer/2000 repository, generate SQL*Loader scripts which will load data from legacy extract files into the corresponding table. Use the **Generate Loader Script** program to generate scripts for either a specified table-level mapping or for all mappings.
5	Determine a sampling strategy, by which legacy system experts will extract a subset of records from the legacy databases for use in the preliminary data cleansing. Produce extract files as specified. Transfer these extract files to the development system.

Continued on next page

Overview (continued)

Step	Action
6	Run the generated DDL scripts to define legacy tables and the SQL*Loader scripts to load the tables with extracted legacy data.
7	Based on the legacy-target mappings, target validations, and mapping transforms, generate data cleansing analysis scripts for each table-level mapping. Use the **Generate Cleansing Script** program to generate scripts for either a specified table-level mapping or for all mappings.
8	Run the generated cleansing scripts. Each script examines the extracted data, to determine which column-level legacy data violates the mappings and transforms specified. A report is generated for each script run. These reports are then printed to yield preliminary data cleansing results.
9	Review preliminary data cleansing results, to determine where dirty data exists and how it will be fixed. Update table and column level mappings and transform rules as appropriate. Write cleansing programs on the legacy systems if required.
10	Based on the legacy-target mappings, target validations, and mapping transforms, generate conversion scripts for each table-level mapping which is not handled by an Oracle-provided interface file. Use the **Generate Conversion Script** program to generate scripts for either a specified table-level mapping or for all mappings.
11	Based on a detailed analysis of conversion requirements, write any additional programs or data definitions that cannot be generated automatically, and apply any required customization to generated programs. Anticipated hand-coding includes views to SELECT DISTINCT rows from specified tables; and PL/SQL scripts, when SQL does not provide the capabilities required to perform a particular conversion. View and code definitions will be stored in the Designer/2000 repository and generated.
12	Based on the legacy-target mappings, target validations, and mapping transforms, generate conversion reports for each table-level mapping which is not handled by an Oracle-provided interface file. Use the **Generate Conversion Report** program to generate programs for either a specified table-level mapping or for all mappings.

Continued on next page

Overview (continued)

Step	Action
13	Using either the previously defined sampling strategy or a refined version of the strategy, for example, to examine more legacy data, produce another set of extract files to be used in the conversion. Transfer these extract files to the development system. Transfer the data from the extract files to the Oracle database load tables via the previously generated SQL*Loader scripts.
14	If required, regenerate the cleansing scripts to incorporate any additional mappings and/or rules specified as a result of the review of data cleansing results performed in step 9. Run all cleansing scripts on the new versions of the extract files, to verify that legacy system cleansing and/or revised mappings and rules have been applied correctly. Generate the cleansing reports and distribute for analysis.
15	Ensure that a target database has been created to hold the results of the conversion. For custom conversions, run the conversion scripts, to transfer the data from the load tables to the specified target tables. For Oracle-provided interfaces, run the interface programs.

4. 2 Software Inventory

Overview

The Data Cleansing and Conversion Support Application consists of the proprietary software programs listed below.

Program Title	Usage
Legacy Tables Report	List, by legacy application, all legacy tables, specifying whether they scheduled to be converted.
Legacy Table Mapping Report	List, by legacy application, all legacy tables scheduled to be converted, and the target table(s) to which each is mapped.
Target Table Mapping Report	List, by target application and table, the target table and the legacy table(s) from which the table is specified to receive data. This report will help pinpoint missing data and/or mappings.
Legacy-Target Table Mapping Screen	Enter or maintain legacy-target table-level mappings, as well as comments about each mapping.
Legacy Column Mapping Report	List, by legacy table and column, all legacy columns scheduled to be converted, and the target column(s) to which each is mapped.
Target Column Mapping Report	List, by target table and column, the legacy table and column(s) from which it is specified to receive data. This report will help pinpoint missing data and/or mappings.
Legacy-Target Column Mapping Screen	Enter or maintain legacy-target column-level mappings, as well as comments about each mapping.
Generate Loader Script	For each legacy table scheduled to be converted, generate a SQL*Loader script to load data from the extract file into the corresponding Oracle table.
Generate Cleansing Script	For each legacy table scheduled to be converted, generate a SQL script to examine the data loaded and report on violations of Designer/2000-specified validations or conversion transform rules.

Continued on next page

Overview (continued)

Program Name	Usage
Generate Conversion Script	For each legacy-target table mapping, generate a SQL script to convert the legacy data into the specified Oracle table. The script will apply all validations and transform rules specified in the repository and column-level mappings.
Generate Conversion Report	For each legacy-target table mapping, generate a program to report on the conversion of legacy data into the specified Oracle table. The script will report on all rows in the legacy table which were not successfully converted, and the reason for the failure, that is, which validation or transform rule was violated.

4.3 Using the Data Cleansing and Conversion Software

Overview

This section gives general rules for using the data cleansing and conversion software.

Push Buttons and Menu Options

All data entry and inquiry screens use a standard menu bar. The menu bar includes most actions needed to enter and review data. In addition, push buttons have been included at the top of the screen to facilitate screen usage. These buttons provide shortcuts for actions which are also available on the menu bar. Usage of these buttons is summarized below.

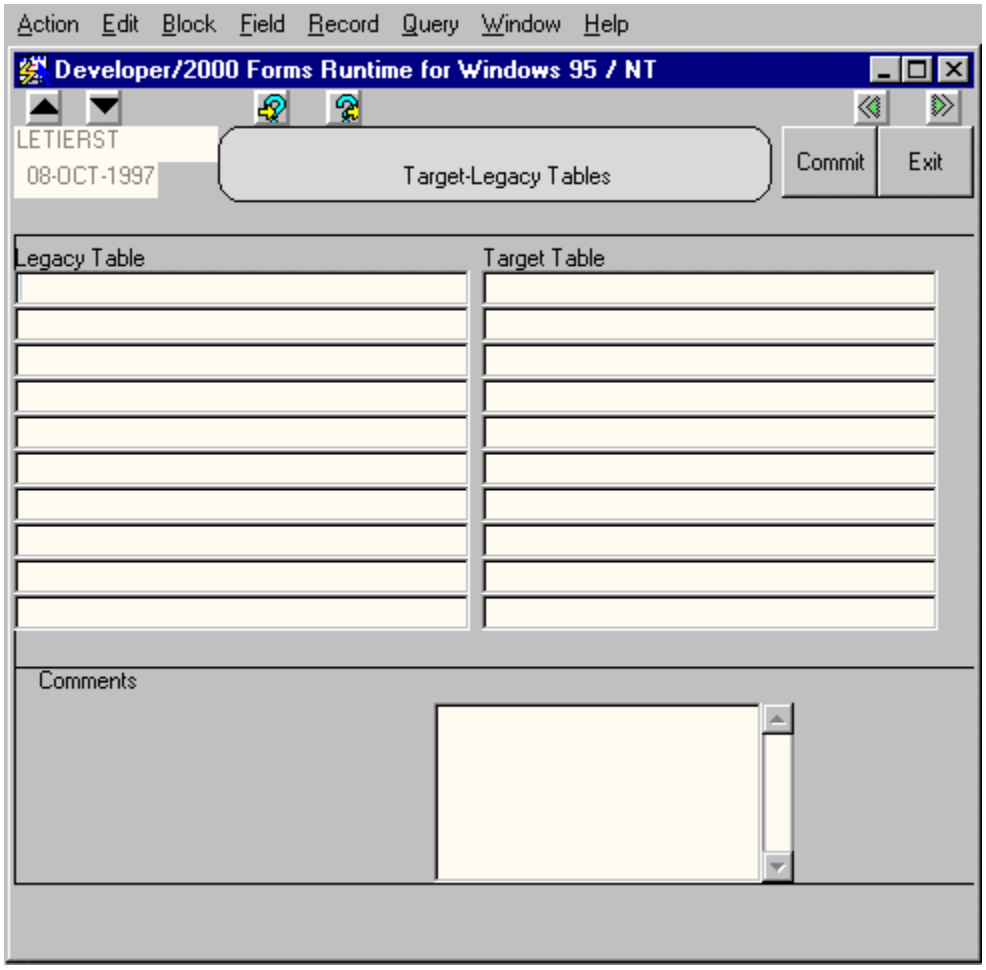
Button	Action
	Return to the previous record in a multi-row block.
	Advance to the next record in a multi-row block.
	Enter query mode.
	Execute a query to retrieve legacy-target mappings based on the currently entered query criteria.
	Used on multi-page forms to navigate between pages.
	Commit any outstanding transactions.
	Exit from the program. If you have any outstanding transactions, you will be asked if you want to commit them.

4.4 Legacy-Target Table Mapping

Introduction

Mapping a legacy table to one or more target tables is the first step is designing and implementing a data cleansing strategy.

Screen Display



Continued on next page

Procedure

Follow the procedure below to enter legacy-target table mappings.

Step	Action
1	Start the program by running the form on Phase2: H:\appbulletin\tab10020.fmx. Enter your user name and password and connect to the database CPMD.
2	To review data previously entered, execute a query to retrieve the records of interest.
3	To enter a legacy table name, you may either type the name, or press List-of-Values (F9) to view a list of all legacy tables, find one or more of interest, and select the desired table. Legacy tables are restricted to the applications CPICS, CAPS, CENTRAK, PACMAN, and CBS.
4	To enter a target table name, you may either type the name, or press List-of-Values (F9) to view a list of all target tables, find one or more of interest, and select the desired table. Target tables are restricted to the Oracle applications PA, PO, AP, AR, AOL, and GL, and the custom application CPMS-LDM (logical data model). If a table name seems to appear more than once (as it may, because of inconsistencies in the CPMS_LDM application), select the first one on the list and/or the one which is spelled corrected (PA_TASKS, not PA_TASKSES).
5	Enter a comment. At a minimum, enter your initials and the date the mapping was made. You may also enter comments, questions, open issues, or other appropriate remarks.
6	Press [Commit] at judicious intervals to save your work.
7	Once entered, a mapping cannot be modified (although the comment can be changed). To modify a mapping, delete the original by locating it and using the menu entry Record->Remove, and then enter the replacement.

Continued on next page