“Desktop Data Warehousing”

An alternative approach to traditional data warehousing projects that leverages R&R Report Writer on a desktop PC.

A White Paper From Liveware Publishing

By Daniel Levin

© Copyright August 2001

Overview

In the past several years, there has been a trend in databases toward accumulating data into a "Data Warehouse". Usually this process involves moving and processing large volumes of data onto a large-scale platform such as Oracle or SQL Server, and storing that information on very large servers with huge amounts of storage and retrieval capacity.

An industry has evolved around this methodology, and several large technology companies have embraced it as a high-value application of their equipment and software. Companies that build the equipment for this purpose and software companies that manipulate the data that results from the effort have grown tremendously. These companies promise unlimited value to the enterprise under the heading "Business Intelligence". They propose large value from "data-mining", on-line analytical processing (OLAP), and "data-mart" one-stop shopping for information, and also promise complete data security.

The core purpose of any data warehousing effort is clear: allow people in the enterprise to put their data to use in marketing, sales, operations, planning, decision support, customer service, management control and so on. This is always the primary function of any information system, not just the data warehouse. While there is little doubt that some value comes from data warehousing efforts, the current approach advanced by technology companies, and supported by many enterprises' Information Technology departments, is not the only methodology to fulfill these objectives. In fact, the current methodology for data warehousing is often the most expensive and difficult one!!

At Liveware Publishing, we have been pursuing a different concept that we have found works better and costs much less. Our methodology should have significant appeal to any enterprise that falls into these categories:

- Organizations that are too small to provide the resources that the complex systems, software and support a traditional data warehousing project demands, but still wants the benefits such an effort promises.

- Organizations whose data and operations are too diverse or decentralized to make a traditional data warehouse effort feasible.

- Organizations under time constraints that make the lengthy data warehouse development process unappealing.

- Organizations concerned about the initial and ongoing costs of a traditional data warehousing effort.

- Organizations that have already invested in a traditional data warehousing effort, but find it still has gaps due to a changing business environment, acquisition of new business units or critical data sources that are not easily integrated with the existing data warehouse.

In short, nearly every organization can utilize our methodology, but with little additional investment in either information technology or software, and with far less development (and hence a much shorter deployment interval) than a traditional data warehouse.

We realize that our proposition runs counter the trends in this industry and, in particular, the thinking in many IT departments. Nevertheless, we are prepared to defend our position in theory and by example; we can deliver the required results to operating department in far less time, far less money, and with far greater flexibility than traditional data warehousing.

Nature of the Data Problem

The fundamental purpose of a data warehouse project is to prepare raw data for distribution to operational departments that generate and then utilize that information. By its very nature, the data in a data warehouse is for 'reporting', which we will broadly define as any form of extraction. The data tables in the warehouse are not for dynamic additions, transaction processing, editing or spontaneous reference; such functions are reserved for production databases tied to specific application software.

To build a traditional data warehouse, one or more copies of the production databases are made to a central location or locations, often with a variety of processes performed on the data tables. These processes include one of more the following:

- Standardization of all data tables on a single platform, to allow for single-tool manipulation of the data into its final form.

- Translation of specific data fields to a common coding basis – such as identification of all 'people' by their social security numbers, instead of a variety of coding schemes.

- Consolidation of several identically or similarly structured data tables into a single table containing the common information.

- Recasting raw data into normalized or de-normalized data tables, based on the context of how that raw data is likely to be used.

- Accumulation of raw data into time or group related totals – such as adding sales totals by region or year from individual sales transactions.

- Creation of metadata units and/or result sets by joining data tables based on their common values or contextual relationships.

Figure 1 presents a schematic of a traditional data warehouse under this generalized function, including icons that represent end-users of the information.

Naturally, these complex processes require a significant amount of effort to design and implement. In general, a traditional data warehouse will take several months to design and even longer to implement due to one overriding problem: contextual uses of the raw data.

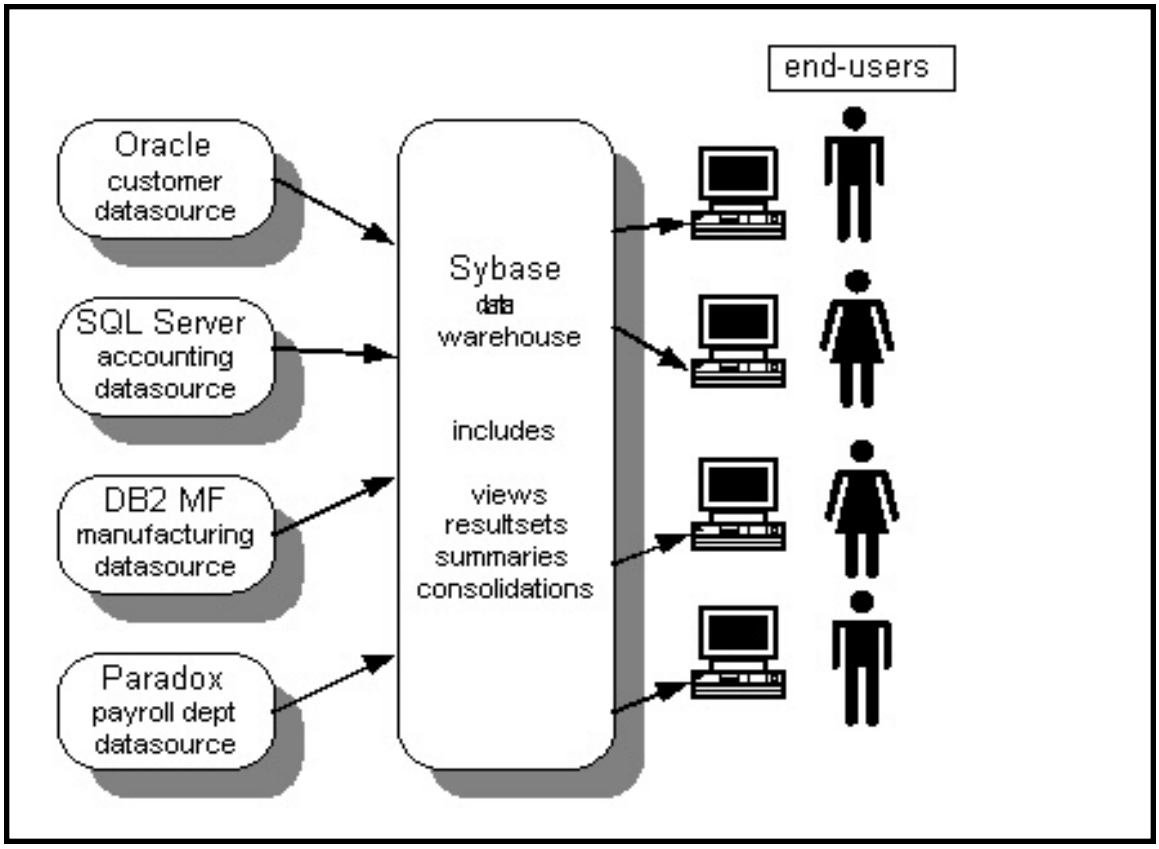

Figure 1 -- Schematic of 'traditional' data warehousing effort.

Collecting and Storing Data vs. Using It

Collecting data is relatively easy and getting progressively easier. Passive measurement and data collection systems – including sensors, digital cameras and, especially, bar-code scanners and similar devices – can build huge volumes of accurately coded data. Computer entry and database sharing make building large databases a much less onerous task than in any time in history.

Similarly, electronic devices for quickly recording, storing and retrieving this raw data have also improved exponentially, and are likely to continue to do so for some time. Microprocessor speeds continue to double every couple of years and storage devices can read and write the data bits in ever-larger quantities. Networks to transfer these bits from one location to another have also improved in speed and 'bandwidth' – the ability to send a certain amount of data electronically from point A to point B in a unit of time – is expanding so quickly that its overabundance is universally believed to be just a matter of time.

What has not changed very rapidly, if one argues it has changed at all, is the purposes – or 'contexts' – in which that data is used. The core functions of any organization that relies on its information – accounting, sales & marketing, customer service, production (and/or delivery of services), resource management (including inventory, personnel administration, distribution) – have not changed nor are they likely ever to change. In addition, the manner with which these functions are performed, in relation to the information available, are not very much different than in the pre-computer age.

With the advent of computerized information, management can access data in greater detail and analyze that raw data to replace gut instinct in management decision-making. Experience and wisdom still rule, however, often trumping the analysis of the data. Since management is often about anticipating the future, the collection and analysis of raw data, which only reflects the past, must only aid in decision-making, not replace it. Similarly, information can be communicated to others using modern technology, but decisions regarding who, what, where, when and why still require management input. The same issues are involved in the use of information for routine and special administrative tasks – auditing, follow-up, etc. – need intelligent human control.

Traditional data warehousing projects have attempted to address the end-users needs as diagrammed in Figure 2 (page 6). This effort usually starts with in-depth interviews with the end-users to identify the types of information they need to extract from transactional systems, and the timing and form in which the data must be delivered. Then the information technology (IT) department, often in concert with consultants in data warehousing, proceeds to develop all of procedures, result sets and, often, the final output form for delivery to the end-user of that information.

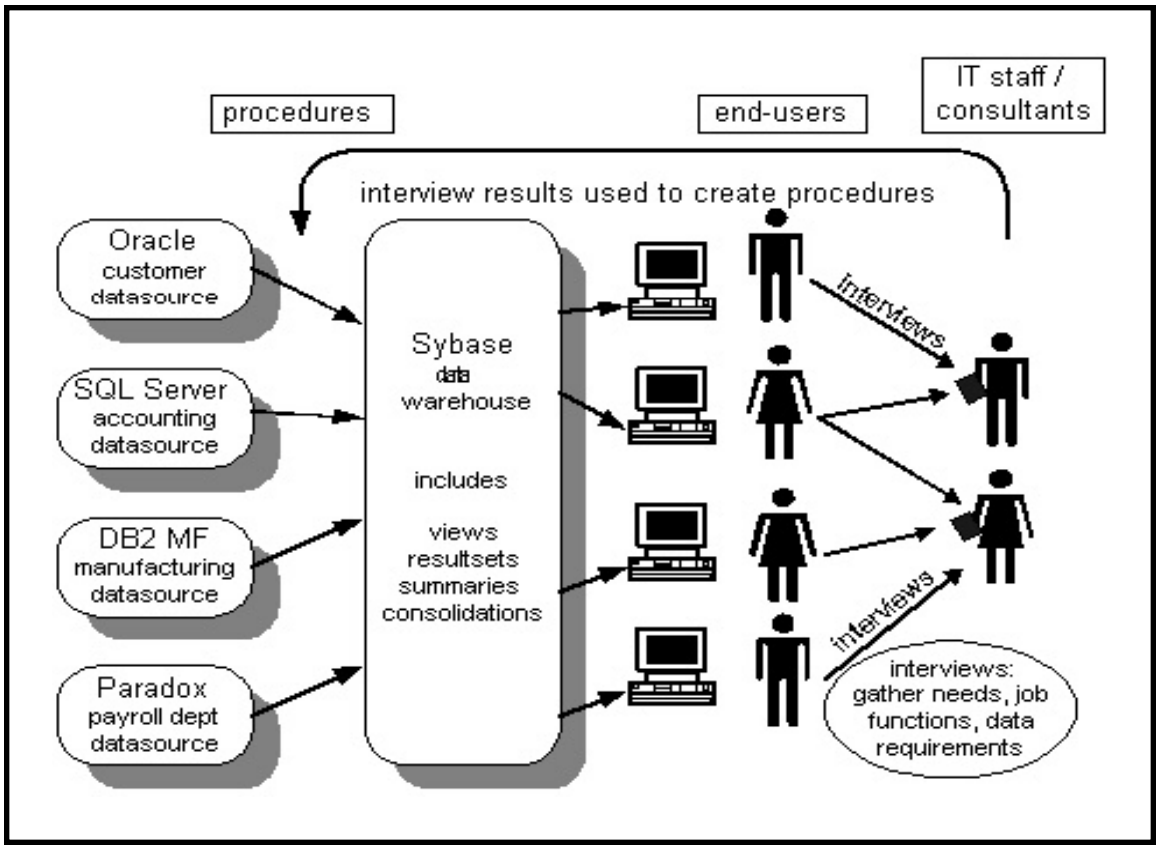

Figure 2 -- IT Consultants' involvement and information transfer in a traditional data warehousing effort.

This has become a popular methodology, but the turn-around time involved, and the amount of knowledge transfer from personnel in operational departments to those in IT needed to produce viable results is extensive – and expensive. Often, it takes consulting firms knowledgeable in both IT *and* operational requirements to produce initial results. It then becomes necessary to maintain the consultants or add IT staff to develop new or modified final results as circumstances change.

Limitations of a Traditional Data Warehouse

There is no way to escape the need for the people who use the information to have some modicum of control over how the information is organized. As their circumstances change, even slightly, the methods with which they will need to extract data from transactional applications programs will change. In addition, in nearly all cases the people have unique knowledge of their circumstances. That knowledge may be either stored in a local data source, such as a spreadsheet or PC/network-based database, or simply within their own minds. Traditional data warehousing projects cannot address any of these elements, or require such a high level of maintenance rendering the entire effort moot or prohibitively expensive.

As stated above and diagrammed in Figure 2, the knowledge transfer for the operational functions can be achieved, but only with the ongoing intervention of

IT departments, and/or hired consultants. The most common rationale for this method is that people in operational departments do not understand the database schema nor the tools with which to manipulate the data into its final form. (We will put aside, temporarily, our objection to that premise.) Yet, those building the procedures, result sets, interim and final output in the data warehouse project cannot always know, nor anticipate, what is in the mind of the person in the operational department.

Expertise, managerial "gut instincts" and other elements of data analysis and routine uses are hard to describe to an IT person or consultant who hasn't faced the same challenges day after day. Even if a data warehousing project could do so, the availability of the consultants or IT staff to address, prioritize and complete the myriad requests from end-users in operational departments in daunting.

Perhaps an even greater challenge to traditional data warehousing is its inability to address local data sources. By their nature, data warehouses can only summarize and relay the information that is in them. But suppose – and this is a ***very*** common situation – that the end-users have some combination of the following:

- A spreadsheet that contains special information, such as analysis factors, for which there is no corollary in any table in the data warehouse.

- Data transmitted from a third party in raw form (e.g., phone records from the long-distance service provider).

- A departmental database or "best of breed" application running within the LAN or even local PC environment.

- End-user specific databases, either built by the end-user or legacy applications designed to address mission-critical (or hard-to-replace) functionality.

Adding any of these data sources to the data warehouse, just to incorporate them in the final result, is highly problematic and costly to the point of impossibility – in a traditional data warehouse.

Note: We believe that the 'Desktop Data Warehousing' is analagous to desktop publishing, where pre-press work was moved substantially from the professional graphic artist realm to the typical end-user who has better than average computer knowledge. Some understanding of fonts, graphics, page layout fundamentals and so forth was a prerequisite, but end-users embraced it due to greater control and quicker turnaround for their complex documents.

Rationale for a Desktop Data Warehouse

Our thesis is this: since it is the end-user – most often a person in an operational department, not in IT – who:

a) needs the information to perform their tasks,
b) understands those tasks,
c) understands the actual data (as opposed to, perhaps, the data structures or schema), and

d) has a local computer readily available

then it is at the end-user's computer – the desktop – at which one should construct the data warehouse.

Note: Our experience is that end-users will ultimately define the success or failure of the data warehousing project, and many of those projects do fail.

In order to do what we propose, our approach need not address all of functions of a traditional data warehousing project, as outlined in the section entitled "Nature of the Data Problem" – but it would be nice!! Nor does our approach eliminate the need for IT departments and consultants to participate in the process. It simply reduces the amount of knowledge transfer necessary to produce final results, and makes some of that knowledge flow ***from*** IT departments and consultants to the end-users. This is a crucial difference, as end-users set priorities and the timing for completion of final results, not IT departments. In addition to end-users explaining their needs to IT and consultants, these consultants prepare components for end-user to manipulate, and explain database schemas and the method for using the tools.

Now we challenge the premise that end-users in operational departments don't (or can't) understand database schemas and data management tools. For the first item – database schemas – the end-user doesn't have to know all of it, just those portions most applicable to their needs. IT and consultants can also develop metadata, templates, samples and other short-cuts to assist the end-user; this is a far better and cheaper method than training IT and consultants to do everybody's job. For the second item – end-users not understanding the tools – we patently disagree. Well-designed tools – such as R&R Data Warehouse – are no more complex to use and understand than a spreadsheet program.

Training, practice, and mentoring from IT and consultants are necessary, but with the added benefit that the end-users also learn more about data management. They also begin to appreciate that data quality is paramount; they can't use what wasn't input – and input correctly. The importance of coding, consistent entry, timely updates and auditing procedures all carry extra weight when the end-user sees the data coming together in the final results.

Creating a Desktop Data Warehouse with "R&R Data Warehouse"

Our "Desktop Data Warehouse" is diagrammed in Figure 3 (page 9). Note that local data sources are treated just like every other. Our diagram also makes full use of any existing data warehouse – this is just more raw material for desktop manipulation. (This is also why a Desktop Data Warehouse might not need to address every item outlined above.)

When we gained control of R&R Report Writer in 1999, our target was to produce a desktop data warehouse software program that mimics all of the functionality of a traditional data warehouse. With R&R V9.0, we have succeeded in our goal. In the sections that follow, we will demonstrate how R&R can perform these core functions.

Standardization of Data to a Single Platform

For the first of our data warehousing core functions – platform standardization – we do so using DBF data files stored in a location from which the desktop PC can readily access them: a local or network drive. 'DBF' (a.k.a. xBase) table structures have been around for many, many years and many PC applications either use them directly or can output them as an export format. DBF tables are also true structured data files, as opposed to plain or delimited text, XML, HTML or text files.

Note: This added granularity also protects against data corruption. Individual tables are quickly recoverable or not subject to any corruption when other tables are damaged in disk failures or bad sectors. This is not the case when recovering the entire database in MDB or other formats.

One clear advantage of DBF files over some other choices is that each table is its own file, and can therefore be identified, copied, updated or otherwise referenced within the file structure separately from any other table. This makes DBF files easier to work with for reporting, since tables are the fundamental unit of reporting. Compared to using, say, Access MDB databases, DBFs are more versatile in a data warehouse environment due to the individual storage of each table.

Each DBF table has a header at the beginning of the file that identifies the sizes and types of data fields, number of records, and so forth. This gives direct context to the continuous stream of data that follows the header, which can be of virtually any length or size. DBF tables allow for very long record length, character field lengths of up to 254 characters, numeric fields with explicit decimals, true

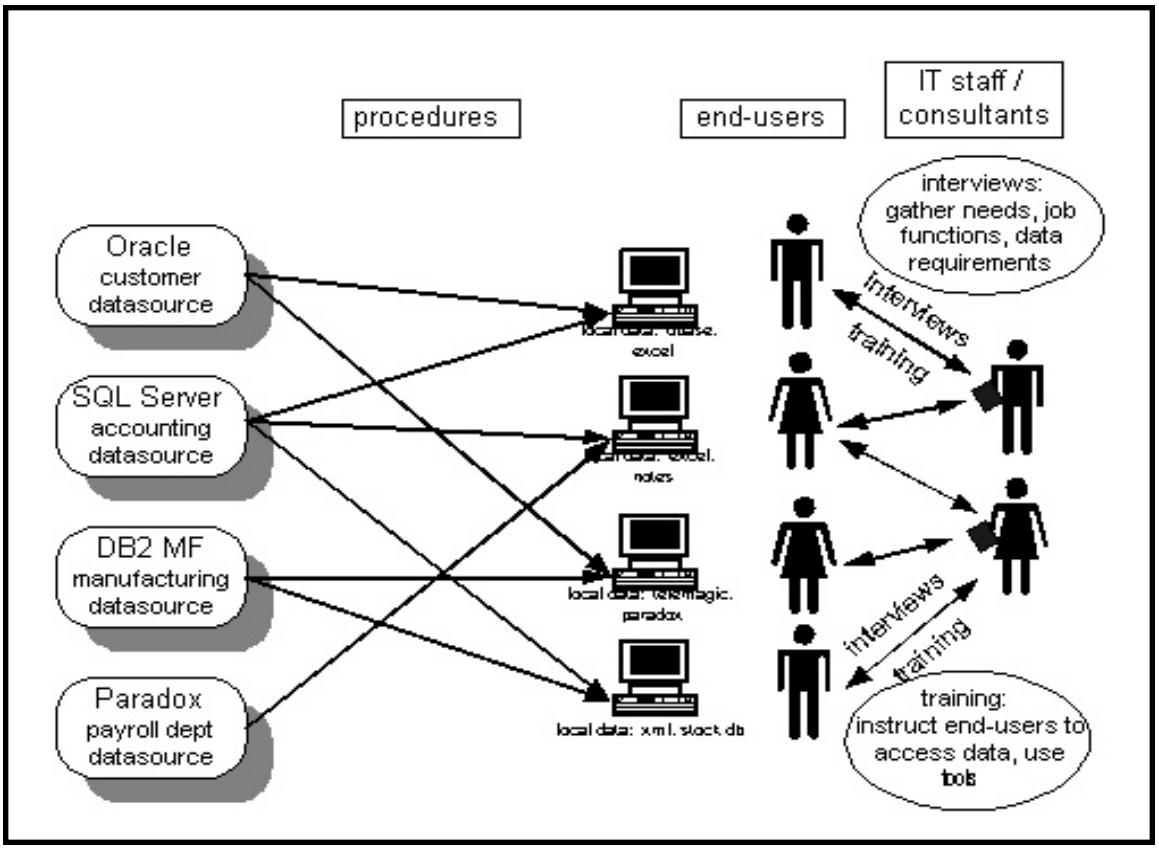

Figure 3 -- Schematic for a 'Desktop Data Warehousing' effort, with information transfer between end-users and IT flowing both ways, and data going directly to desktop computers.

date / datetime fields, and true logical fields. Each field is handled based on its assigned type, and these types represent the fundamental data types that reporting uses across data systems. For example, transformation of text- or numeric-stored dates is not necessary in DBF files.

In addition to all of the above reasons, with R&R Data Warehouse one can index the DBF files since DBF files allow for many types of indexes to exist on them without being defined within the table itself. For data maintenance and production data systems, this has been seen as a drawback of DBF files. For reporting, however, it is an *extreme* advantage. Data warehouses, by their nature, are not production systems so re-indexing tables can be performed without restrictions such as locking out others. There are also many inexpensive utility programs that will read DBF files and perform any necessary maintenance functions without taking the files off line for any reporting or extraction needs.

For a complete description of these advantages, refer to the forthcoming Liveware Publishing white paper "*Universal Join Technology and Limitation of SQL Joins*".

In traditional data warehousing projects, Oracle, SQL Server, Sybase (as indicated in our diagram) or some other database has often been the choice for standardization. (See Figure 2.) While these databases are equally valid, it takes a great deal more effort to populate them than it takes R&R to build DBFs via a desktop data warehouse. In addition translating local data sources, such as a spreadsheet or departmental application, to a large platform database is an involved process; Lotus, Excel and many other PC programs can directly output to a DBF file, saving at least one step in the data warehousing process.

Note: R&R can export any band you specify; more about that in the section below entitled "*Accumulate and Summarization Contexts*".

Using R&R Data Warehouse the standardization process simply requires that you specify that the R&R report should produce a DBF (xBase) file extract in place of a printed report. R&R will automatically create the file header and populate the DBF file, assigning the field length and type specified in the report.

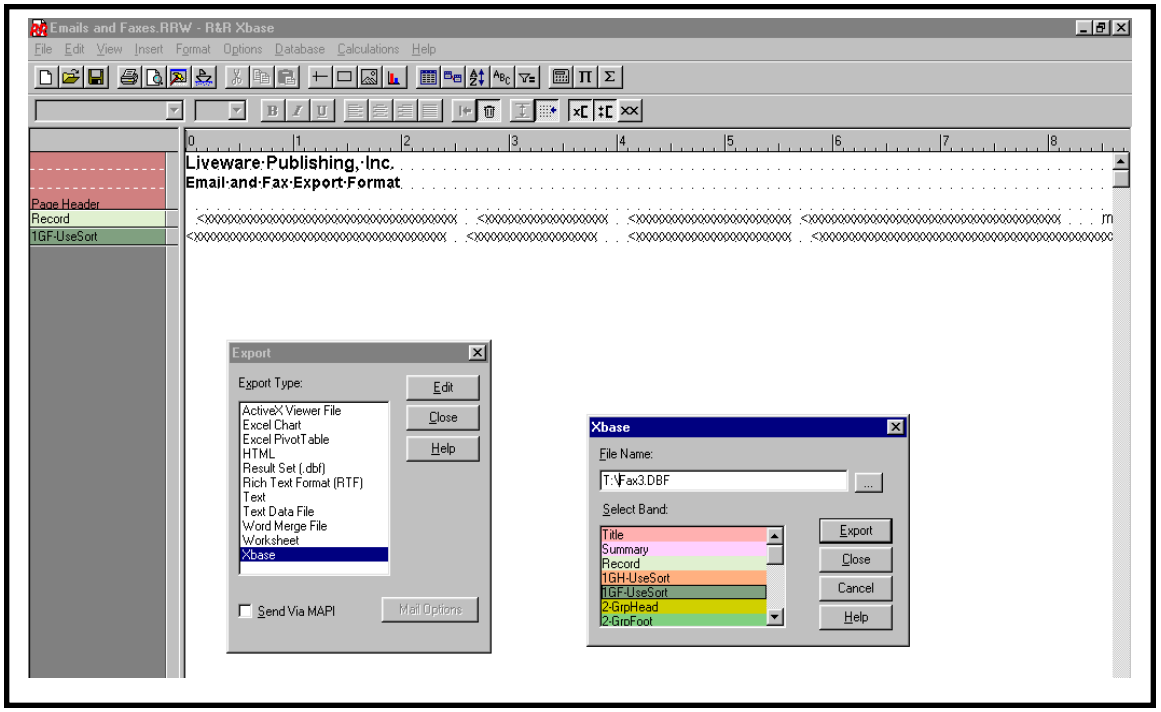

Figure 4 -- R&R xBase export options with band selection.

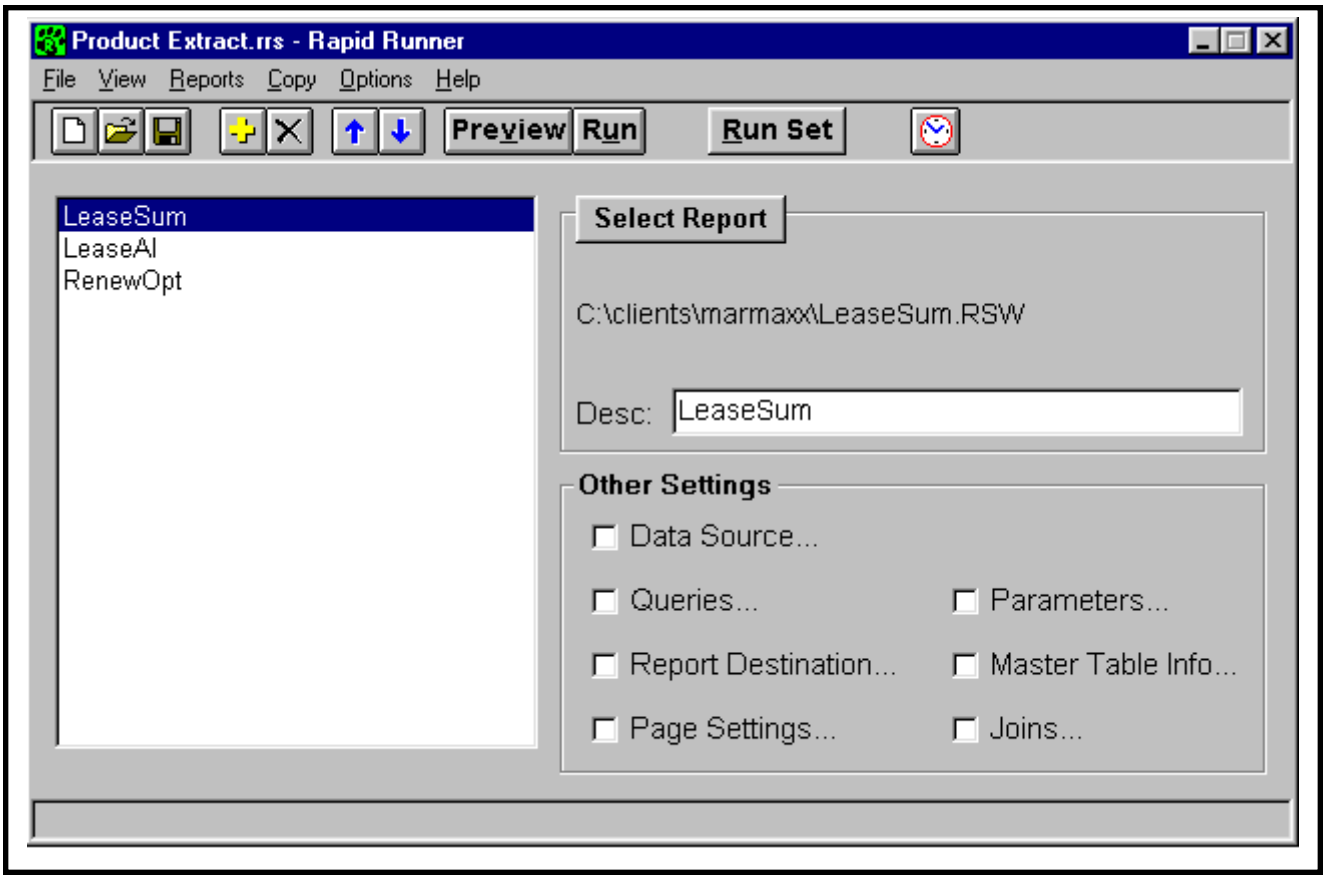

Figure 5 -- Rapid Runner 'batch' of reports to extract a series to a series of data tables in DBF format. Any of these reports may be run independently, or all of them as a set.

Figure 4 shows a typical xBase extract report format, with overlaying Export dialog box.

Each R&R report designed in this manner would produce a separate data table for the data warehouse. To produce a series of data tables, one would use Rapid Runner to create batches of these types of reports, as shown in Figure 5. Note that each of these 'extract' report formats may have a different 'data source' designation; one report could point to an Oracle database while the next on the list could be a SQL Server data source.

Within Rapid Runner one may specify the 'Data Source' and 'Destination' as shown in Figure 6 (page 12), or R&R will use the settings saved with the actual report. This means that one may modify all elements of this data extraction on the desktop, and that any combination of these may be saved, copied, transferred or modified among all the Desktop Data Warehouse users. Create a set of extract reports, then the report 'set' – the fundamental unit of Rapid Runner – can be defined and forwarded with the individual reports to anyone who needs to use it. Those reports or just the Data Source and/or Destination can be modified and saved, or simply executed once. Therefore, the modular design of the routines to build standardized data tables in R&R Data Warehouse offers superior flexibility to traditional data warehousing methods.

Data Translations in a Desktop Data Warehouse

One of the most difficult aspects of combining and reporting across data sources is that they have often evolved at different times and in different places. This has resulted in contextual differences, particularly as these differences apply to coding. Take, for example, a bank that merges with two others, one of which was the result of the merger of five smaller banks. If each bank has maintained its own account and customer databases, there will be eight separate systems, until all of the existing data and applications can be ported to a common system. (Also, what happens if yet another merger is planned?)

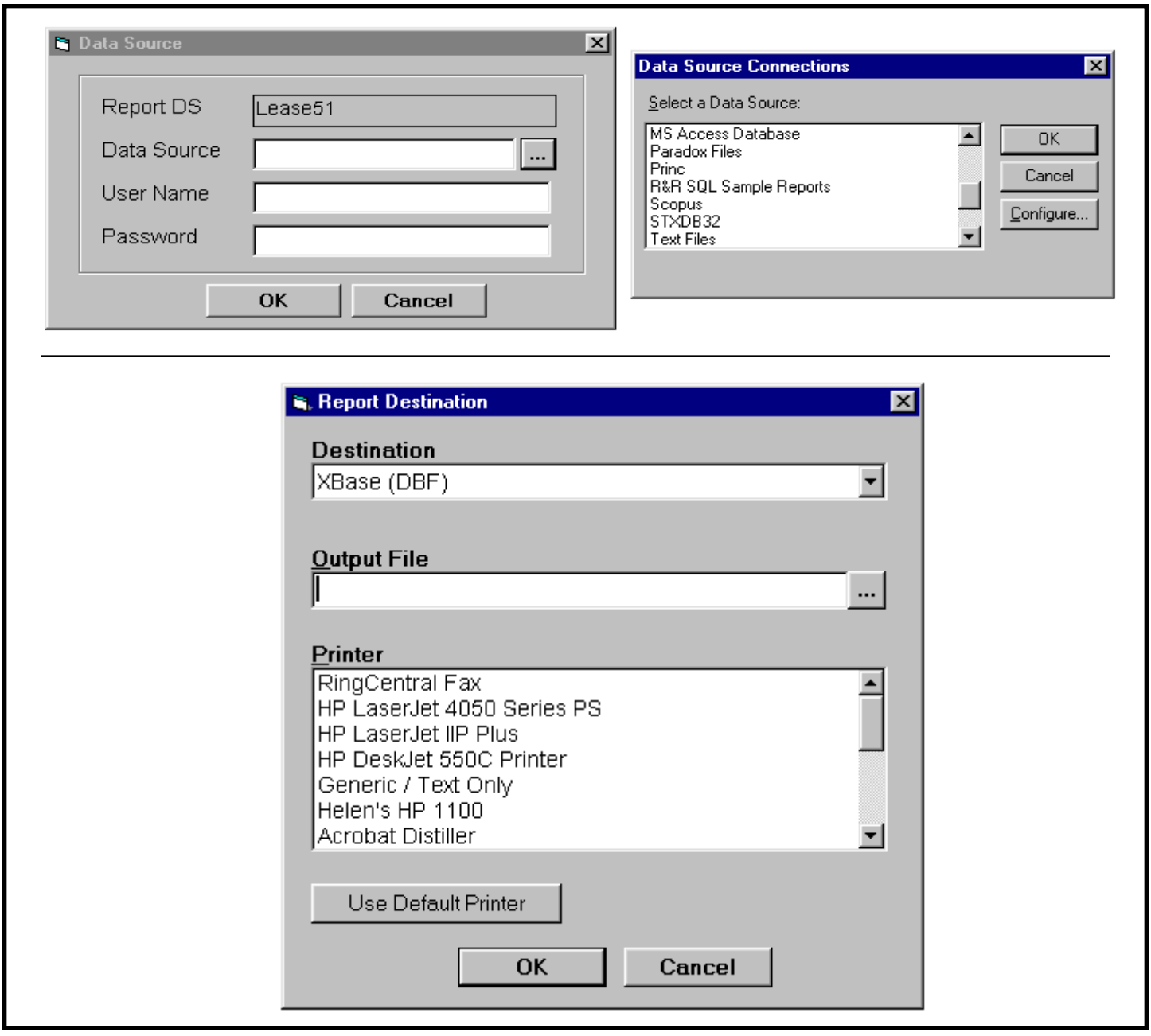

Figure 6 -- 'Data Source' and destination are both variables that can be set, stored, run or overridden in Rapid Runner. This provides a very flexible mechanism for extracting data whenever to and to wherever needed.

Traditional data warehousing projects have attempted to manage these difficulties by devising translation programs and "stored procedures" to produce translated data tables that produce common elements across all of the production data structures. While a valid effort, such a methodology is limited since the contextual difference will multiply faster than they could be addressed in building those procedures.

A core function of any reporting tool is to handle a variety of contexts, which can change rapidly or vary widely from case to case and location to location. Therefore, our strategy (as reflected in R&R Data Warehouse) is to let the raw data stand on its own as much as possible, and address the context when R&R generates the final report output. When convenient the data extractions can incorporate common, recurring translation schemes very easily, and may be modified as circumstances or contexts change. This is not a requirement, however, as the report format the produces the final output can include virtually any necessary translations.

This methodology has several advantages. First, additional data sources – resulting, for example, from the acquisition of another small bank – are quickly translated with no modification of existing extracts. Second, R&R lets you use the raw data, when necessary, so that the end-user can define a final report without modification of the data extracts.

Figure 7 demonstrates one application of R&R's translation methodology. In this example, we are attempting to find out which customers of Bank 1 have accounts with Bank 2 as well. "Bnk1Cust" is the table that holds customer information for Bank 1's account holders. This file's primary key is social security

Bnk1Cust

SOCSECNO	FNAME	LNAME	FIRSTOPEN
387-93-0449	PAUL	WILSON	07/02/1998
038-33-2991	SCOTT	FITZGERALD	03/27/1997
485-30-3002	WILLIAM	HAKIM	12/11/2000
129-03-9763	PENELOPE	DUBOIS	09/30/1999
376-38-2043	PETER	ADAMSON	04/12/1998

Bnk2Cust

SSNUM	FIRSTNM	LASTNM	OPEN1ST
263097746	ALICE	STEVENS	09/02/2000
387930449	PAUL	WILSON	07/02/1998
38332991	SCOTT	FITZGERALD	03/27/1997
114160375	JANICE	YABLONSKI	06/18/1999
376382043	PETER	ADAMSON	04/12/1998
45887635	THOMAS	SULLIVAN	11/18/1997

Figure 7 -- Data structures for two asynchonous, parallel customer tables for Banks 1 and 2.

number (field SOCSECNO) that is 11 characters long, and therefore includes the dashes. "Bnk2Acct" is the account information table from Bank 2. The difficulty is that the legacy system at Bank 2 also uses social security number (field SSNUM), but stores it as a 9-digit numeric value with leading zeros trimmed.

This does not represent a problem for R&R. In fact, we can perform the join one of two ways: 1) define a calculated field that produces the numeric equivalent of SOCSECNO, or 2) define an index expression that turns SSNUM into a punctuated character string. (We should note here that SQL does not support either of these methods. Both fields' values must be stored with the commonality to produce a valid JOIN.) Figure 8 displays the second method, with a fairly extensive but not terribly complex formula to produce the index expression from SSNUM. It was this capability, added in R&R xBase V9, that gives 100% flexibility to join tables and completes R&R's exclusive Universal Join Technology.

Consolidation of Data from Disparate Data Sources

One of the most obvious functions of a data warehouse is, as the term implies, the storage in a single location – and perhaps a single set of data files – of data maintained in multiple locations. As in other warehousing functions, such as distribution and trucking, data warehousing for consolidation serves a very valuable function. A traditional data warehousing produces most of its value in data consolidation, just as a products warehouse does so for distribution functions. A desktop data warehouse has difficulties duplicating these functions but, as will be demonstrated below, can either augment or replace.

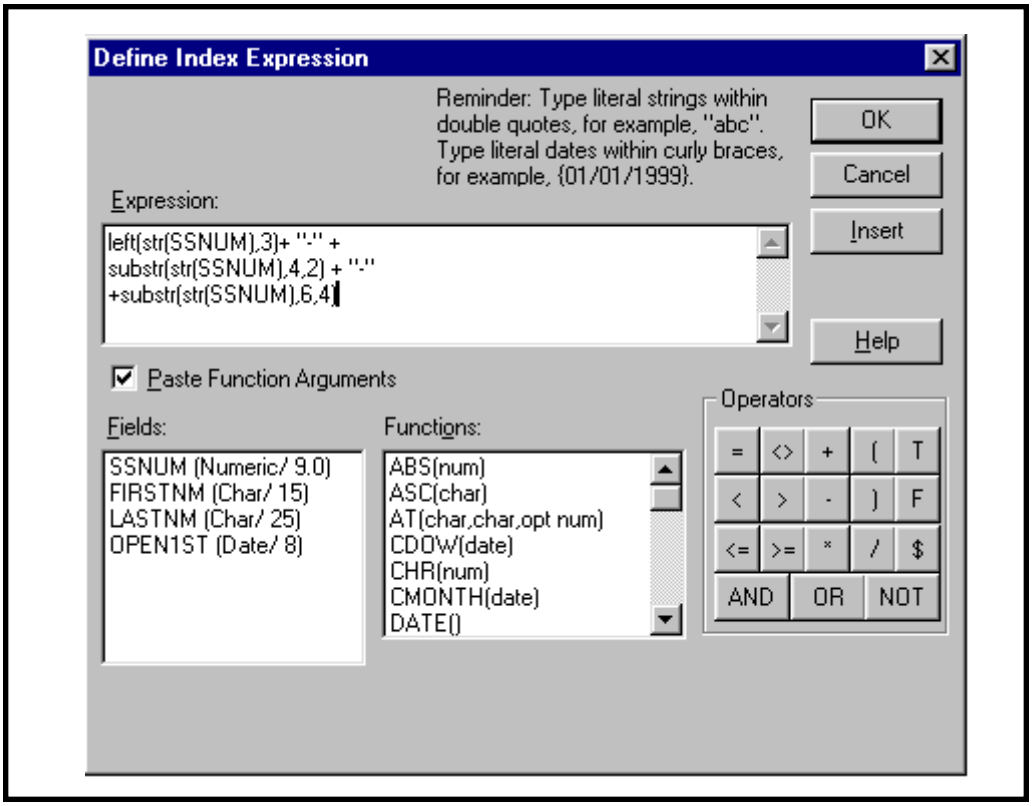

Figure 8 -- Index expression to turn numeric SSNUM to the character equivalent.

There are Desktop Data Warehouse functions that can streamline some consolidation processes and eliminate others entirely. Utilizing these capabilities of a Desktop Data Warehouse can thereby enhance the traditional data warehouse. For example, suppose that a chain of 75 restaurants collects sales data from duplicate systems in each of its locations. Each night every store's computer automatically sends the day's sales, inventory, receipts and other data to a central data repository where the data is accumulated into single tables for each category to allow for regional and national management reporting.

In the above case, it would be unreasonable to expect every regional and national manager with interest in this data to extract the data from all of the stores via a Desktop Data Warehouse. A traditional data warehouse would serve to consolidate data files and then make them available for reporting to anyone who may need that data. Each record's source restaurant would be indicated within the record for identification. This also facilitates reporting within and across stores for the regional and national managers who are monitoring trends or analyzing the data.

Note: A forthcoming update for R&R will provide for this kind of consolidation.

In most traditional data warehousing projects, however, the consolidation efforts would likely go farther and require additional development, maintenance and storage that is not necessary where Desktop Data Warehousing is also employed. For example, creation of time-series data files – such as separate monthly and quarterly archive files for each set of tables, or individual regional sets of tables – is not necessary when a Desktop Data Warehouse model is in place. The key functions eliminated from a traditional data warehouse project are any that further process those data files into interim, consolidated result sets. The Desktop Data Warehouse can handle any of those consolidations and can do so with greater flexibility than a traditional data warehouse.

One particular component of R&R's Desktop Data Warehouse makes data consolidation viable: the 'multi-scan' join. Figure 9 (page 16) demonstrates the simplest form of this result set construction, but more robust joins are allowed, even one that would consolidate all 75 restaurants' data files!! (We don't recommend it for practicical reasons, *not* because the theory behind the multi-scan join or its implementation in R&R is unsound.) In a multi-scan join, two *or more* tables that have a one-to-many (scan) relation to a common parent are joined to that parent in parallel. In the result set, each parent record produces one record for each record found any of the related tables. To extend the example above, if a restaurant has 250 eat-in tickets (records) for a day and 75 take-out tickets in separate data tables, the result set will have 325 records. Every table's fields are represented as a portion of the composite record, with the parent portion always containing data, but only one branch of the related tables containing data at any time.

In effect, the two tables sharing a common parent table are merged together as if they were one table all along, but with different fields being populated depending on the type of record. Figure 10 (page 16) illustrates the same relation,

STORE		EATINTIX			TKOUTTIX		
STNUM	LOCATION	TIXNO	SIZE	AMT	TIXNO	REGNO	AMT
15	Athens, GA	12736	5	47.13			
15	Athens, GA	12737	2	12.73			
15	Athens, GA	12738	4	31.85			
15	Athens, GA	12743	6	41.27			
15	Athens, GA	12745	3	22.02			
15	Athens, GA	12746	4	50.98			
15	Athens, GA	12748	1	12.15			
15	Athens, GA				12739	14B	21.68
15	Athens, GA				12740	18C	57.23
15	Athens, GA				12741	14B	33.64
15	Athens, GA				12742	13A	15.47
15	Athens, GA				12744	13B	18.73
15	Athens, GA				12747	14B	43.99
15	Athens, GA				12749	18C	41.16
15	Athens, GA				12750	18C	38.94
15	Athens, GA				12751	14A	17.35

Figure 9 -- "Multi-scan" result set for 'eat-in' and 'take-out' tickets for restaurant number 15.

but in a slightly different way. Instead of a wide range of empty fields as one branch is populated at a time, the fields from the two tables are intermixed. This doesn't look as disjointed as in Figure 9, but the result set is 100% equivalent. In fact, this structure for populating data tables occurs regularly in applications software where different fields are populated at different times.

Note in Figure 9 that eat-in tickets and take-out tickets have slightly different fields structures. Eat-in tickets have a server ID number (referencing another data table containing the server's identity) and take-out tickets have a packaging

STNUM	LOCATION	TIXNO	SIZE	AMT	REGNO	TIXTYPE
15	Athens, GA	12736	5	47.13		EATIN
15	Athens, GA	12737	2	12.73		EATIN
15	Athens, GA	12738	4	31.85		EATIN
15	Athens, GA	12739		21.68	14B	TKOUT
15	Athens, GA	12740		57.23	18C	TKOUT
15	Athens, GA	12741		33.64	14B	TKOUT
15	Athens, GA	12742		15.47	13A	TKOUT
15	Athens, GA	12743	6	41.27		EATIN
15	Athens, GA	12744		18.73	13B	TKOUT
15	Athens, GA	12745	3	22.02		EATIN
15	Athens, GA	12746	4	50.98		EATIN
15	Athens, GA	12747		43.99	14B	TKOUT
15	Athens, GA	12748	1	12.15		EATIN
15	Athens, GA	12749		41.16	18C	TKOUT
15	Athens, GA	12750		38.94	18C	TKOUT
15	Athens, GA	12751		17.35	14A	TKOUT

Figure 10 -- Identical result set to Figure 9, presented as if the two ticket types had been stored in the same data table all along.

cost value. In this case, the consolidation is taking place across *asymmetrical* data tables, in contrast to the consolidation of the 75 restaurant's tables described above. When a R&R Desktop Data Warehouse is in place, the only consolidation necessary in a traditional data warehouse is for a large number of multiple *symmetrical* (identical) data tables, and then only for convenience.

Figure 11 presents R&R's Related Tables dialog box (only applicable in R&R xBase Edition), that demonstrates the multi-scan join. First, one would define the two one-to-many (scan) relations from the STORE (restaurant) table

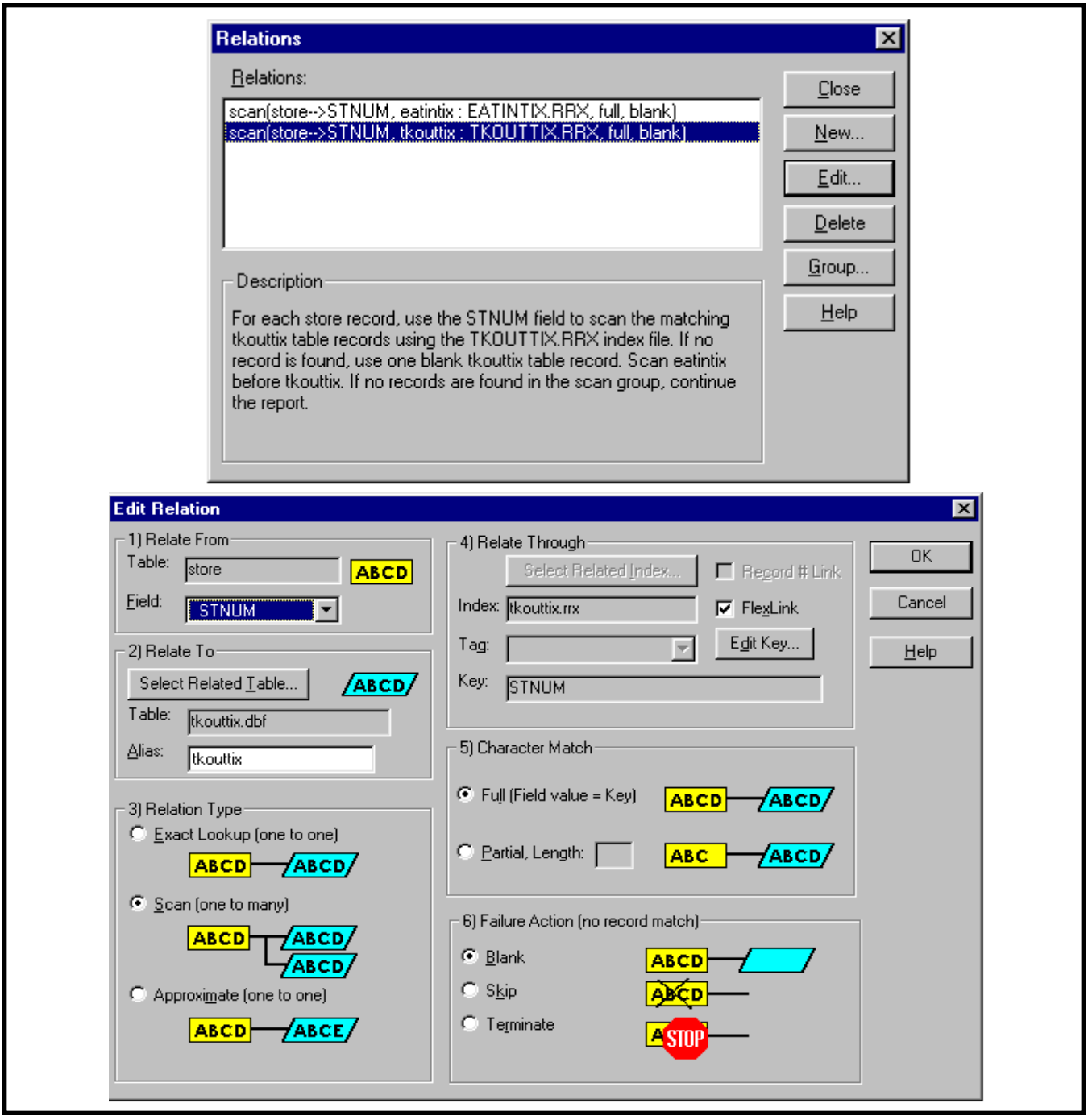

Figure 11 -- R&R xBase Edition Related Tables and relation definition dialog boxes to produce the result set in Figure 9 via a "multi-scan" join.

to EATINTIX and TKOUTTIX transactions tables. R&R automatically recognizes that this is a multi-scan relationship, as evidenced by the activated action button labeled 'Group'. The relations to EATINTIX and TKOUTTIX are grouped in parallel since R&R sees that they share the common parent table with separate scan relations.

This is a crucial difference between R&R and any SQL-based reporting tool. No other reporting tool (including R&R SQL Edition!!) will automatically recognize this construction as a unique join form among tables. SQL language can replicate this common contextual relationship between tables (often known as the parent/child/child), but only with an extensive series of commands that no SQL reporting tool has been able to simulate to date. A multi-scan join is *not* the same as a UNION JOIN (even with a COALESCE in SQL-92), as this SQL join is still, fundamentally, between two tables while multi-scan is a join of three or more tables. In R&R, the designated 'parent' need not be a single table; the only requirement is that the parent is a) a single table or b) two or more tables joined with a one-to-one relation.

Multi-scan relationships in context are very common. Here are just a few general examples:

- reporting across 'current' and 'archive' transaction tables
  Ex. sales by quarter analysis from active table for the current year and archive table for prior years

- reporting from a table in two or more contexts in a single result set
  Ex. split commission statement between 'Salesman A' and 'Salesman B' for the same sales transaction

- consolidating data from multiple sources
  Ex. Income statement from multiple subsidiary companies

- summarizing disparate categories of events across a shared entity
  Ex. Employee benefits statement with paid time off plans, health and welfare plans, legislative plans (FICA, Workers comp), and pension plans from separate tables

In effect, the multi-scan join builds a result set that allows for independent, but related, branches within it. Further, R&R recognizes each branch's independence, but provides for summarization across these branches as the context requires. SQL languages and reporting tools that rely on it (including R&R SQL) do not contain any branching controls. Therefore, any context that requires it almost always demands the creation of a series of interim result set and the development of procedures to prepare the necessary result set. This can be done, of course, but the expense related to that effort is far greater than allowing R&R's multi-scan join to do the work.

R&R contains all of the functions to merge fields across either symmetrical or asymmetrical tables that share a common context. For example, as shown in Figure 12, we would want to merge the eat-in and take-out AMT field values to a single column for totaling, averaging and so on. (We use this as our example because there are many contexts in which this would be a plausible reconstruction on the raw data.) To achieve the new column of data, we would simply create a calculated field in R&R as shown in Figure 13. The SCANNING() function allows us to designate the multi-scan branch and return the value from a field (or formula result) for that branch. These SCANNING() functions can be nested for multi-scan relations among more than two related tables.

The result set in Figure 9 and 10 are consolidations of two tables, but a ten- or fifty-table consolidation follows the same conventions. As one might conclude from the formula in Figure 13 (page 20), keeping tracking of so many branches by the user becomes a daunting task (though not for R&R itself). We recommend that this methodology for consolidation be kept to a maximum of half a dozen or so branches. This works fine for all of the examples sited in the bullets above, but would not be appropriate for the example of 75 restaurants with symmetrical data tables. This methodology does work well for the case where 50 of the restaurants had one system and the rest had another. The traditional data warehouse would make two sets of consolidations, then R&R would handle the final consolidation as needs and contexts dictated.

Restructuring Raw Data for Normalized and De-normalized Tables

Tables within applications programs are maintained in a variety of data structures designed primarily to store information for retrieval within the application.

STORE		EATINTIX			TKOUTTIX			
STNUM	LOCATION	TIXNO	SIZE	AMT	TIXNO	REGNO	AMT	*TXAMT*
15	Athens, GA	12736	5	47.13				47.13
15	Athens, GA	12737	2	12.73				12.73
15	Athens, GA	12738	4	31.85				31.85
15	Athens, GA	12743	6	41.27				41.27
15	Athens, GA	12745	3	22.02				22.02
15	Athens, GA	12746	4	50.98				50.98
15	Athens, GA	12748	1	12.15				12.15
15	Athens, GA				12739	14B	21.68	21.68
15	Athens, GA				12740	18C	57.23	57.23
15	Athens, GA				12741	14B	33.64	33.64
15	Athens, GA				12742	13A	15.47	15.47
15	Athens, GA				12744	13B	18.73	18.73
15	Athens, GA				12747	14B	43.99	43.99
15	Athens, GA				12749	18C	41.16	41.16
15	Athens, GA				12750	18C	38.94	38.94
15	Athens, GA				12751	14A	17.35	17.35

Figure 12 -- New column for the result set based on the value of* AMT *in either* EATINTIX *or* TKOUTTIX *table.

This retrieval process can be performed to either the screen – for reference or editing purposes – or via reports. For most applications, by far the most common purpose for data retrieval is for reference or editing of that data and, hence, retrieval to the screen takes precedence in the design of the data structures. After all, people working with the data to record or monitor individual entities or transactions are using the data continuously.

Take as an example the role of a customer service representative (CSR) at a travel agency. A CSR would regularly check the computer program for the status of customers' ticketing, account information, preferences, and so on. This could happen dozens of times every day and the CSR would do so on the screen and the information about the customer or reservation would be retrieved from the stored data within the reservation application program.

With this priority in mind, most application programs' data structures are designed for simple retrieval to the screen. Accumulations of data elements for common functions – such as those I described above – are maintained within the same data record so that the computer can quickly retrieve that set of information. Work activities – referencing an open reservation to answer a customer's question over the phone, for example – are much more time sensitive than an analysis of occupancy that a hotel manager might perform on the same type of data. Therefore, the data structure of the application program would best be designed to serve the common, time-sensitive function.

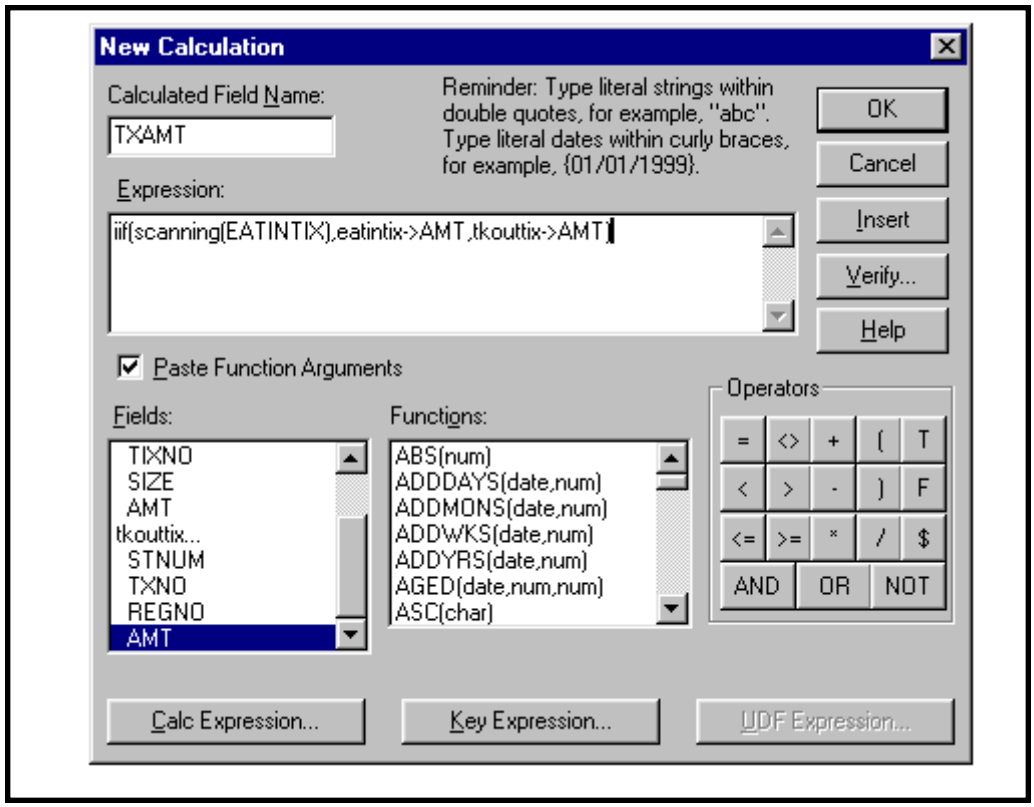

Figure 13 -- New Calculation dialog box to define the column named TXAMT from Figure 12. This calculation formula uses R&R xBase Edition SCANNING() function to specify which branch of the multi-scan join to test.

In our current example a typical (abbreviated) data structure for a reservation application might look like Figure 14. There would be one table for the customers' identities, one for their reservations, one for their account activity, and maybe another for their preferences – including credit card numbers, room options and other data. In Figure 14, the credit card information is "normalized" in that each credit card number has its own record.

Another equally valid possibility for this application is where the customer file contains fields for each customer's three credit card preferences, since most people have only this number or so primary credit cards. This arrangement is shown in Figure 15 (page 22). Such a data structure is known as a "de-normalized" structure since credit cards 1,2 and 3 are within the same record, rather than individual records. This data structure allows for simpler retrieval to the screen for an individual, since this credit card information is in the same record as other information about the customer. In the normalized data structure of Figure 14, the application would likely allow an unlimited number of credit card entries for each customer, and would therefore need a separate screen of information to display the list and allow the CSR to select the record in question for reference or editing.

This situation represents the fundamental trade-off between normalized and de-normalized data structures. Normalized structures provide greater flexibility and more information storage, while de-normalized structures are simpler and more direct. While not getting into a debate over which methodology is better, the current trend in application design is toward normalized structures since they are more robust. This does, however, increase the complexity of an application

CUSTOMER

CUSTID	FNAME	LNAME	LASTTRIP
449	BILL	HILLYARD	07/02/2001
033	ALICIA	GOLDSTONE	06/27/2001
4802	MICHAEL	WOODS	11/11/2000
1297	FRANCES	DELACROIX	12/30/2000
763	DENNIS	GIBSON	03/12/2000

CUSTCARD

CUSTID	CARDTYPE	CARDNUM	CARDEXP	CARDPREF
449	AMEX	328471029384	09/2002	A
1297	MC	394872029384712	07/2003	A
763	VISA	398273492084783	06/2003	B
449	VISA	384726023094837	01/2002	B
033	AMEX	320493827481	11/2001	A
449	DINERS	3948727634012	12/2002	C
763	AMEX	320948572762	04/2002	A

Figure 14 -- Data structures for normalized customers / credit card preferences. Note that the card numbers and names are just made up.

CUSTOMER

CUSTID	FNAME	LNAME	LASTTRIP
449	BILL	HILLYARD	07/02/2001
033	ALICIA	GOLDSTONE	06/27/2001
4802	MICHAEL	WOODS	11/11/2000
1297	FRANCES	DELACROIX	12/30/2000
763	DENNIS	GIBSON	03/12/2000

CUSTPREF

CUSTID	CARD1	CARD1NUM	C1EXP	CARD2	CARD2NUM	C2EXP	CARD3	CARD3NUM	C3EXP
449	AMEX	328471029384	09/02	VISA	384726023094837	01/02	DINERS	3948727634012	12/02
1297	MC	394872029384712	07/03						
763	AMEX	320948572762	04/02	VISA	398273492084783	06/03			
033	AMEX	320493827481	11/01						

Figure 15 -- Data structures for de-normalized customers / credit card preferences with similar information to Figure 14. Note that each customer has only one record in CUSTPREF with up to three cards' info.

program as more tables, screens and programs are needed to manage the extra information and complexity or relationships among the data tables.

Once the data structure has been set, changing it is possible but most uses of the data simply evolve to work within whatever structure is available. Those uses may call for normalized data to be generated from a de-normalized data structure, and vice versa. To address this requirement some systems have adopted a hybrid structure that would contain both the credit card table from Figure 14 *and* the credit fields from the customer table in Figure 15. A series of programs would keep the two files in sync as one or the other was updated, but this method necessarily adds programming overhead.

What does this discussion have to do with reporting and data warehousing? While the normalized/de-normalized structure is typically determined by the most common and time-sensitive uses of the data, these represent a small fraction of the overall uses of the data. In our example, suppose there are 20 distinct processes for referencing, editing, summarizing and reporting from the raw data. The most common referencing and editing functions may constitute 80% of all use activities of the data, they would represent only 20% (applying the well-established 80/20 business rule) of the different kinds of uses – four processes – of the data. That is, for every 100 times the data is retrieved for use, 80 of those events are for four distinct activities. The remaining 20 retrieval events represent 16 distinct activities. The majority of these later categories of activities are retrievals for summarizing and reporting.

In a traditional data warehouse the need to normalize and de-normalize the data structures to address the reporting activities requires significant effort and expense. Each translated data structure is a unique result set that traditional data

warehouses build and maintain from the application program's data structure. But, as the example above shows, the normalized and de-normalized structures that are created in the data warehouse may only be needed a small percentage of the time each is built. That would mean that the effort and expense to rebuild the necessary result set with the required data structure is wasted unless someone actually needs that result set.

Of all traditional data warehousing functions, this is the one best fulfilled by a Desktop Data Warehouse. R&R provides all of the functionality necessary to recast a normalized data structure to a de-normalized one applicable to any particular use, and to normalize a de-normalized structure as the need requires. R&R can do so with greater flexibility and a much shorter development schedule, and since the recast result set is only generated on demand, there is no waste of time and effort to generate them. (Below we will discuss one of the most common de-normalization processes: data summarization.)

Figure 16 shows the Related Tables dialog that creates a normalized credit card table from the de-normalized data structure in Figure 15. The "trick" to

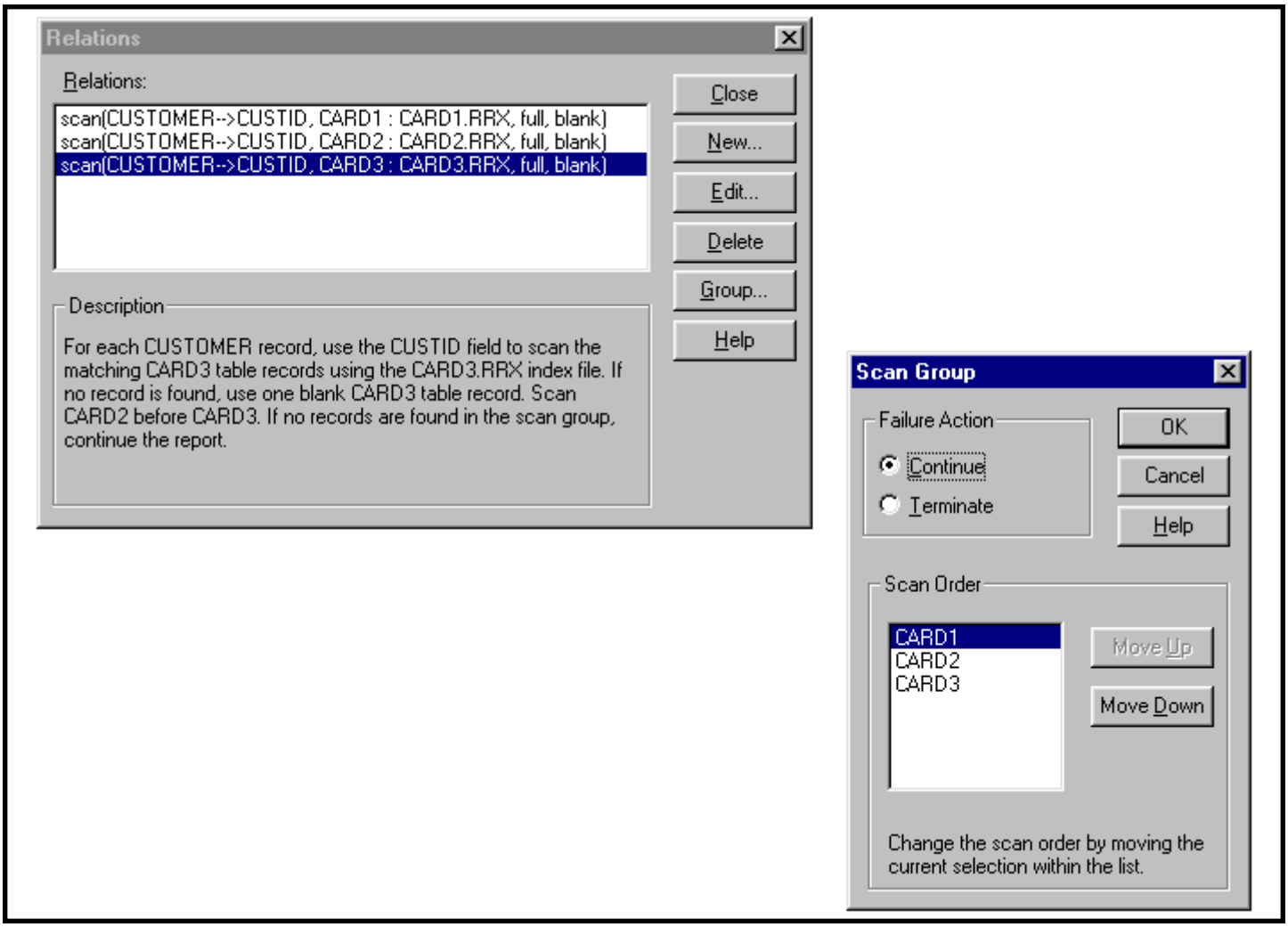

Figure 16 -- R&R xBase Edition Related Tables and Scan Group dialog boxes for the normalization of credit card records from CUSTOMER to CUSTPREF.

achieve this structure is the correct application of the multi-scan join functionality in R&R xBase Edition. (SQL language does not support such a methodology – a severe SQL limitation, by the way – so one must first extract the required data using techniques described in the "Standardization" section above.) In all three joins, the CUSTOMER table is joined to CUSTPREF (utilizing the CUSTID common key, of course). R&R allows this, so long as each table used in the report has a unique alias. For the three related (right side) tables the aliases are *CARD1*, *CARD2*, and *CARD3*, respectively.

In a series of SQL joins on CUSTID, the result set would include just one record, but in R&R xBase we can specify that each join be a "scan" relation. R&R automatically knows to make each scan relation from CUSTOMER to CUSTPREF a separate, parallel branch. The records from each relation are thereby stacked, even though each relation would – necessarily – find just one record each time. In SQL language and, hence, SQL-based reporting tools the ability to distinguish the two contexts is not available. SQL will always use the natural relation between the tables – one-to-one in this case – even though the contextual relationship is one-to-many: one customer to multiple credit card records. R&R immediately recognizes the result set structure and places the three related tables in parallel, as evidenced by the darkened "Group" action button in Figure 16, and the "Scan Group" dialog box also shown in Figure 16.

The remainder of the process simply involves definition of calculated fields using the SCANNING() function to create generalized columns from the parallel branches. Figure 17 presents the one that creates the result set field CARDNUM.

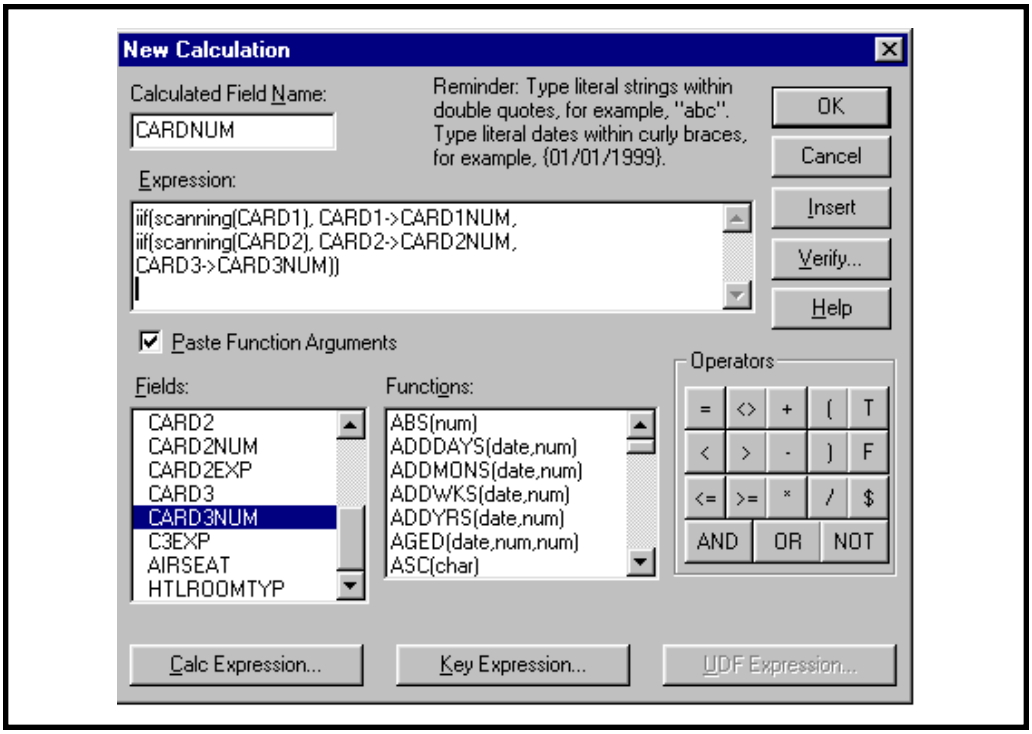

Figure 17 -- New Calculation dialog box to define the column named CARDNUM that will produce the normalized value of the same name in Figure 14 from the data structure in Figure 15.

It is these kinds of contextual relationships, with incongruent but functional data structures, that foil many attempts to streamline reporting. The answer of a traditional data warehouse to overcome them, while viable, is completely unnecessary with the capabilities of R&R xBase and its Universal Join Technology. There are absolutely NO normalization contexts that UJT does not address seamlessly.

Accumulation and Summarization Contexts

Perhaps the most common form of de-normalization – creation of one record from many – is the accumulation of values in records across time or groups. Figure 18 is typical: the raw data of sales summarized by region in the first result set and by month in the second. If we take just the records from the result set that are in bold, we build a new table of de-normalized sales by the new dimensions of region and sales.

SALES (sorted/grouped by REGION)

SALEDT	REGION	PRODLINE	SALEAMT	REGIONTTL
04/15/2001	NE	EXCELSIOR	1150.00	1150.00
03/18/2001	NE	KEYSTONE	850.00	2000.00
04/14/2001	NE	EMPIRE	3948.75	5938.75
		•		
		•		
		•		
05/15/2001	**NE**	**EXCELSIOR**	**1005.00**	**12485.50**
05/12/2001	NE	EMPIRE	545.25	13030.75
03/03/2001	PAC	KEYSTONE	2849.15	2849.15
05/28/2001	PAC	EXCELSIOR	430.00	3279.15
		•		
		•		
		•		
04/22/2001	**PAC**	**EMPIRE**	**1500.00**	**6231.48**

SALES (sorted/grouped by MONTH)

SALEDT	REGION	PRODLINE	SALEAMT	MONTHTTL
03/03/2001	PAC	KEYSTONE	2849.15	2849.15
		•		
		•		
		•		
03/18/2001	**NE**	**KEYSTONE**	**850.00**	**7328.15**
04/14/2001	NE	EMPIRE	3948.75	3948.75
04/15/2001	NE	EXCELSIOR	1150.00	5098.75
		•		
		•		
		•		
04/22/2001	**PAC**	**EMPIRE**	**1500.00**	**11374.00**
05/12/2001	NE	EMPIRE	545.25	545.25
05/15/2001	NE	EXCELSIOR	1005.00	1550.25
		•		
		•		
		•		
05/28/2001	**PAC**	**EXCELSIOR**	**430.00**	**9473.63**

Figure 18 -- Result sets that summarize raw sales data in two separate dimension: REGION and MONTH. Breaks in table indicate where data has been skipped for presentation. Italic record indicates beginning of group and bold indicates the end.

Note: R&R could handle a de-normalization process with 300 (or more) product line values, but that would be impractical and unnecessary. Creation of a new dimension — month-product line – makes far more sense as this result set could always be summarized by month or product line independently.

We can extend this example to one where we de-normalize the monthly sales by three product lines, as indicated in the raw data by three possible values of the field PRODLINE. Figure 19 shows the dialog box with the condition statement to summarize only sales from the "EXCELSIOR" product line to the total field *MoExcSales*.

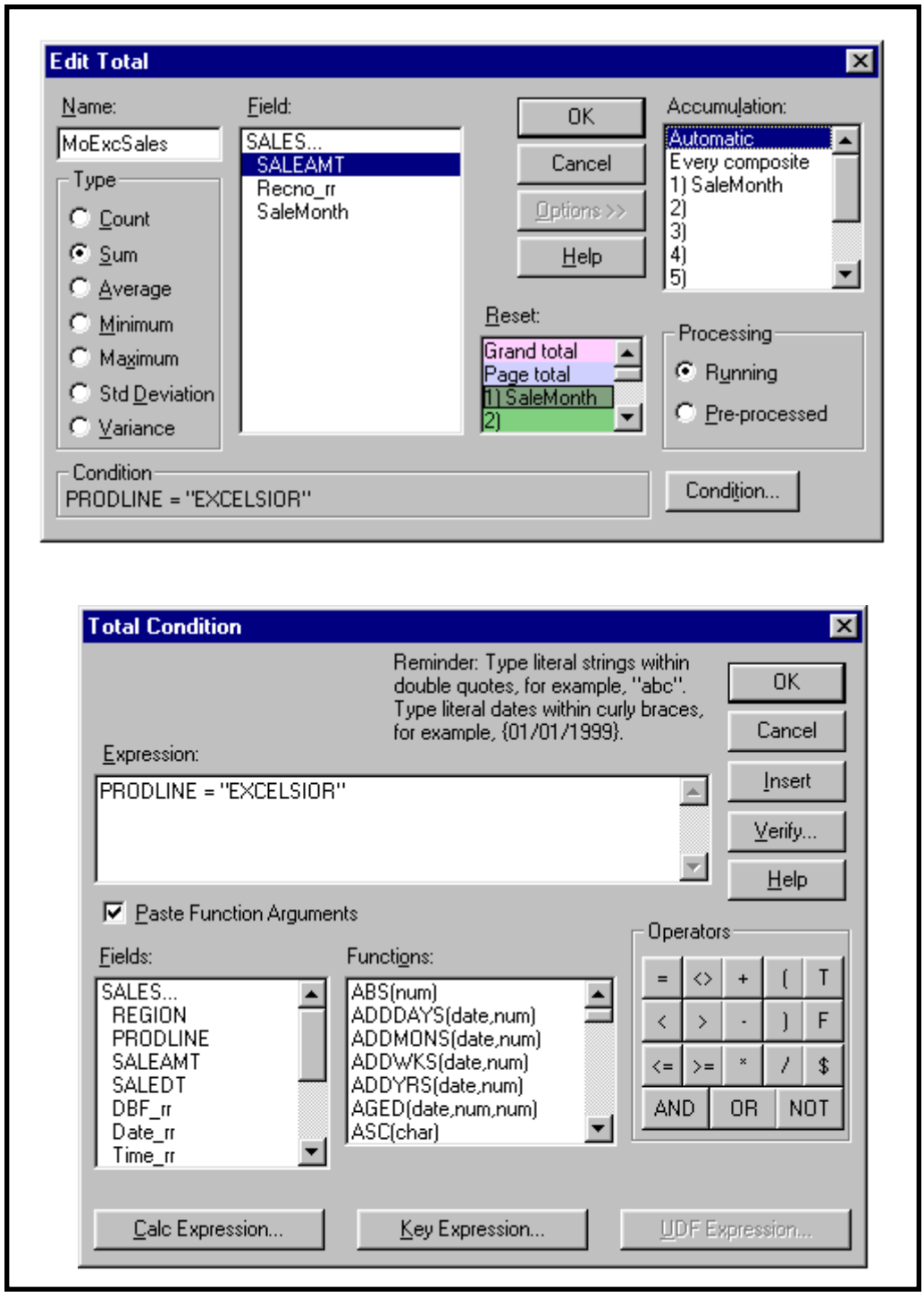

Figure 19 -- Total Field and Total Condition dialog boxes to create a product line sales total within the MONTH dimension result set.

R&R handles this and other de-normalization, summarization, or accumulation processes with equal aplomb to traditional data warehousing. There is one major difference favoring R&R, however. Each time one de-normalizes, summarizes or accumulates data to a new result set – as does a traditional data warehouse – the raw data is lost for those who can only access the new result sets. In a world of ever-shifting contexts, this is unacceptable and attempting to overcome the limitation with a traditional data warehouse is too costly.

Note: Traditional data warehousing projects we've seen attempt to build all the necessary result sets based on the information flow outlined in Figure 2. The cost rises dramatically when the changing contexts require definition of result sets to address those needs.

A Desktop Data Warehouse utilizing R&R does not suffer the same limitation since the same tool that accumulates could return the raw data to the user who needs it. For example, suppose the Analyst who is responsible for analyzing monthly sales in each product line wished to exclude from the analysis a handful of specific items within the product lines. If all this Analyst had available was the accumulated result set – since that is all the data warehousing effort diagrammed in Figure 2 specified – the analysis would be impossible due the existence of unknowns. IT consultants would have to re-interview the Analyst and reprogram the traditional data warehouse to meet this new contextual need. If R&R were accumulating the raw data at the desktop, it would take about 5 minutes to either temporarily or permanently create a new process or replace the former one. With minimal training, the Analyst could do so himself or herself, with no IT intervention and no disruption to anybody.

One more point about normalization and de-normalization in a Desktop Data Warehouse utilizing R&R: one can do both at the same time!! R&R's ability to recast the raw data with normalization and de-normalization techniques allows for ordered or simultaneous processes. Most of the time, a single R&R report with a single set of instructions can accomplish the unique trick of turning the raw data into the required result set and then the final output.

Note: In order to elaborate further, I would have to introduce the concepts of first, second and third derivatives to result sets, so the discussion of when R&R can do so with in a single report will have to wait for another white paper.

Conclusion

My direct experience with hundreds of organizations is that the Desktop Data Warehouse methodology works and is far superior in flexibility and lower cost than traditional data warehousing efforts. The cost ratio is roughly 10 to 1 with the added benefit of almost immediate results and no need to commit to a long term effort without seeing the results and satisfying end users quickly. Ongoing savings are even bigger, particularly when more of the responsibility and knowledge is shifted to the end users. They are the ones, after all, that the data warehouse effort is supposed to serve. They are the ones who understand their work functions best.

It has been pointed out to me on many occasions that end users are "too busy" or "don't want to learn about data" and therefore can never handle desktop reporting or data warehousing. There is some truth to that, but not nearly as much as IT consultants have invested in it. For people whose jobs closely involve information, the knowledge to slice and dice the raw data to administration, communication and analysis duties is a requirement, not a burden. It takes much

more of their time to communication back and forth with IT consultants for every aspect of the data warehousing project than it would be adopt the Desktop Data Warehouse methodology described in Figure 3.

More often than not, the end user are motivated by the results, rather than the process. End users prefer working directly with an IT consultant to produce specific results with quick turnarounds. During those initial and later working sessions, the end users pick up knowledge about the data structures and the techniques to manipulate them. Each R&R report "document" created becomes a template for the next one, and over time the end-user gathers the requisite knowledge to undertake more advanced projects. The IT consultant then takes on a supporting role, providing the expertise for complex issues and managing the process.

Our firm is in the middle of such an effort with a major, national retailer who was frustrated with a new application program for one of its departments. The data structures are much more complex when compared to their previous system and provided much greater flexibility. Extracting the data in the forms they need, however, provided exceedingly difficult, so much so that even the application's developer had abandoned their efforts.

We were invited to show how R&R might address these needs since they had used R&R xBase Edition with their former application. Their new app used a SQL database, so we employed R&R SQL Edition to get started. We soon discovered that SQL joins were not up to the task, even with R&R SQL's inherent capabilities. That left the client and us with three choices:

- Develop the reports utilizing a programming language such as VB to code the output step by step from the raw data.

- Try to use another SQL-based reporting tool, such as Crystal Reports, but likely fall victim to the same SQL limitations.

- Employ R&R Data Warehouse.

The first option was too expensive to consider except as a last resort. It would also require another long delay. We attempted to use Crystal Reports, but it also could not build the result sets we needed. R&R Data Warehouse did the trick, and in just a few hours several critical reports were completed, tested, modified and put into production.

This client has now added several other reports to the project and more are likely to follow. The end users have received training on R&R and have begun to learn where and how data is maintained and in which tables.

I'm quite sure that the Corporation, as opposed to the departments that share this application, would pursue a traditional data warehouse to solve the

same set of problems. But such an effort is just a multitude of smaller projects that could be addressed with a Desktop Data Warehouse.

As I outlined at the beginning of this white paper, organizations with many dynamics could benefit from Desktop Data Warehousing. This client met several of the criteria and knew of R&R prior to commencing the project. They were lucky and made the right choice. Luck doesn't have to be the deciding factor.

About the Author and R&R Report Writer

Daniel Levin in President of Liveware Publishing, Inc., which became the publisher of R&R Report Writer in September 1999. Mr. Levin, his partner, Christian A. Strasser -- who also contributed to this treatise, and associates have worked over the last decade with hundreds of companies through their consulting practice to help those clients use their data most effectively. As an R&R Authorized Trainer beginning in 1993, Mr. Levin has taught over 100 reporting classes and published two books on R&R and the principles of database reporting: ***Relate & Report: Your Guide to Reporting with R&R*** (pub. 1996) and ***The R&R Cookbook*** (pub. 2000).

R&R Report Writer Version 9.0 and R&R Data Warehouse were released in Spring of 2001. More information about R&R and "Desktop Data Warehousing" are available on Liveware Publishing's web site: www.livewarepub.com. R&R is sold throughout the world and has an estimated 500,000 users within tens of thousands of businesses, governments, non-profit organizations and educational institutions. R&R Data Warehouse, combining licenses of R&R xBase and SQL Editions, retails for $700 per concurrent license seat. (Price at publication.)

Notes

This document is © copyrighted material of Liveware Publishing, Inc. It may be copied and distributed freely, but cannot be sold or resold for any form of compensation, without expressed written permission from Liveware Publishing. All rights reserved.

"R&R Report Writer" is a registered trademark of Liveware Publishing, Inc. Screen images from R&R Report Writer and R&R Data Warehouse are copyrighted. "Liveware" is a registered trademark of Liveware, Inc. and Daniel Levin. Other products' names mentioned in this document are trademarks of their respective publishers and owners.