Chapter 15

Multi-Course Dining

Rhonda and her mother Ruth went shopping to buy a high school graduation gift for one of Rhonda's friends. Ruth selected a backpack, but Rhonda was not pleased. She told her mother "How come the gifts we give to my friends are always cheaper than the ones they give to me. Come on, Mom, I have an image to protect."

At the time Ruth disagreed strongly, but Rhonda's challenge did give Ruth pause, and she decided to ask Robert to develop a report from the gift database that sought the answer to these questions:

1) Do the Rubin family children give to their friends more expensive gifts than they receive from their friends?

2) What is the disparity with each friend?

Ruth intends to compare the results for each friend of Rachel, Rhonda, Richard and Randi and seek to reduce the discrepancy in future gift exchanges. Since all gifts given and received are recorded in the gifts database, the answer should lie within the records. Extracting this kind of information, however, is not obvious, but very rewarding when successful.

* * *

Which Fork Am I Supposed to Use?

In order the answer the question Ruth posed to Robert, you must consider that the Rubin family has four school-age children, each with about two dozen friends with whom gift exchanges have occurred. Some friends have had multiple exchanges, while other may have had just a couple.

We could develop a report that summarized gifts exchanges for one Rubin child, then run the report four times, changing the child in question each time. Using the concepts discussed in this Volume, we know that the composite database for such a report would have to look like Figure 15-A.

There are several key components of this table. The first is the grouping based on the calculated field *ExchFriend*, based on the formula:

IIF(RECIP_CD="RHONDA",GIVER_CD,RECIP_CD)

with the query limiting the records to where Rhonda was either the giver or recipient. There are two conditional totals with reset level based on *ExchFriend*: one for Rhonda as the recipient (Ex*RecipVal*) and the other for gifts given (*ExGiverVal*). These two figures would

Summarizing the report for each of Rhonda's friends is a straight-forward sorting/ grouping problem. We solve it easily by defining a field that returns the proper code -- recipient or giver -- for each gift exchange partner. Then we sort, group and derive totals based on the field: *ExchFriend*.

Then we sort, group and derive totals based on the field: *ExchFriend*.

GIFTS.DBF

GIVER_CD	RECIP_CD	VALUE	*ExchFriend*	*ExGiverVal*	*ExRecipVal*
RHON	JFED	50.00	JFED	50.00	0.00
JFED	RHON	15.75	JFED	50.00	15.75
JFED	RHON	21.00	JFED	50.00	36.75
JFED	RHON	12.50	JFED	50.00	49.25
RHON	JFED	18.00	JFED	68.00	49.25
RHON	JFED	11.50	JFED	79.50	49.25
JFED	RHON	25.00	JFED	79.50	74.25
RHON	JFED	19.00	JFED	98.50	74.25
JFED	**RHON**	**30.00**	**JFED**	**98.50**	**104.25**
JILB	RHON	27.50	JILB	0.00	27.50
RHON	JILB	26.00	JILB	26.00	27.50
RHON	JILB	22.50	JILB	48.50	27.50
JILB	RHON	16.00	JILB	48.50	43.50
				⋮	
JILB	**RHON**	**10.00**	**JILB**	**90.25**	**99.50**

Figure 15-A: Composite database summarizing Rhonda's gift exchanges.

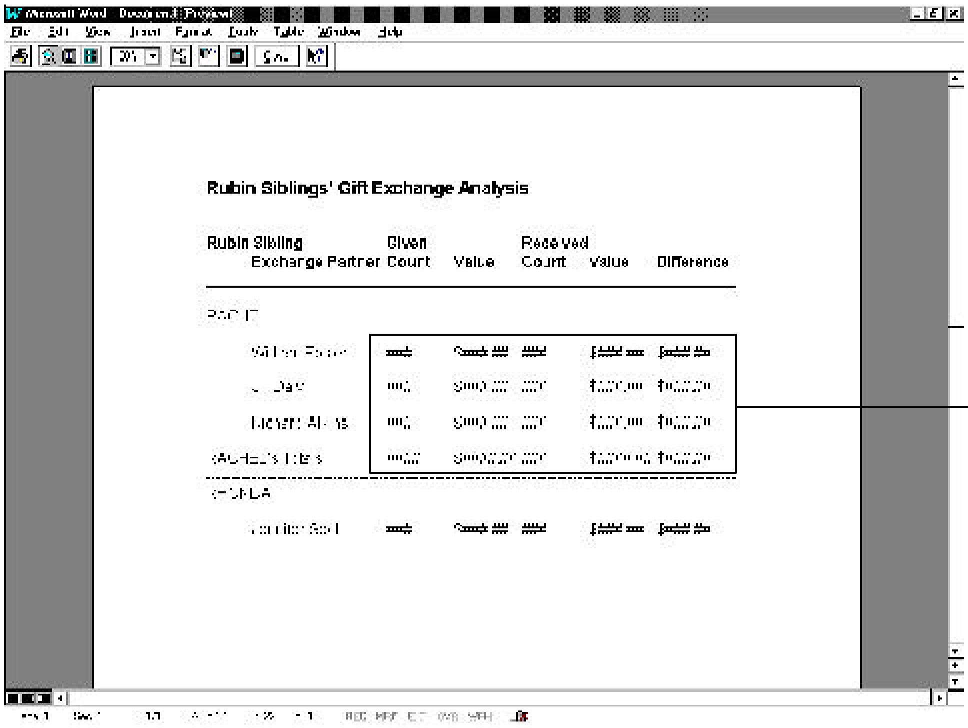
be placed in the *ExchFriend* group footer, thus addressing the report's purpose, but only for Rhonda.

Ruth could change *ExchFriend* and the conditions on the two totals for each of her siblings, add the overall totals (*GTRecipVal* & *GTGiverVal*) and address the entire stated purpose. This method works, but only because there are just four Rubin children as variables. In essence, we are attempting to build a matrix as shown in Figure 15-B. Each time we run the report, we complete another pair of columns under each Rubin child's name.

But suppose Ruth had 50 children!! We would have run the report 50 times. For this example, 50 children in a family is, naturally, a ludicrous possibility. Yet if the database held gift exchange records for all of the children in an elementary school, and we wanted to perform the same analysis as Ruth wants, this technique fails the test. Perhaps a more concrete example of the problem might help our understanding of the problem.

Airline Food

A typical airline timetable, like the one I'm looking at right now for Southwest Airlines, lists each destination city and all of the other



Rubin Siblings' Gift Exchange Analysis

Rubin Sibling / Exchange Partner	Given Count	Value	Received Count	Value	Difference
[illegible]					
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]					
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Manual production of this simulated report, via a spreadsheet or word processing program, is not difficult if we must produce the report for just the four Rubin siblings.

Many people generate reports like this using tools other than R&R or other report writers, and the effort is substantial. R&R provides a better way.

Figure 15-B: Matrix cross-tabulation between Rubin children and gift exchanges with their friends.

cities to or from where one can travel. (See Figure 15-C.) Like our gift exchange issue, each flight has an origination city and destination city. In the time table the same flight from Albuquerque to Baltimore could be listed under the “To” section for Albuquerque and the “From” section for Baltimore. Does this mean that there are two flight records in a database? No.

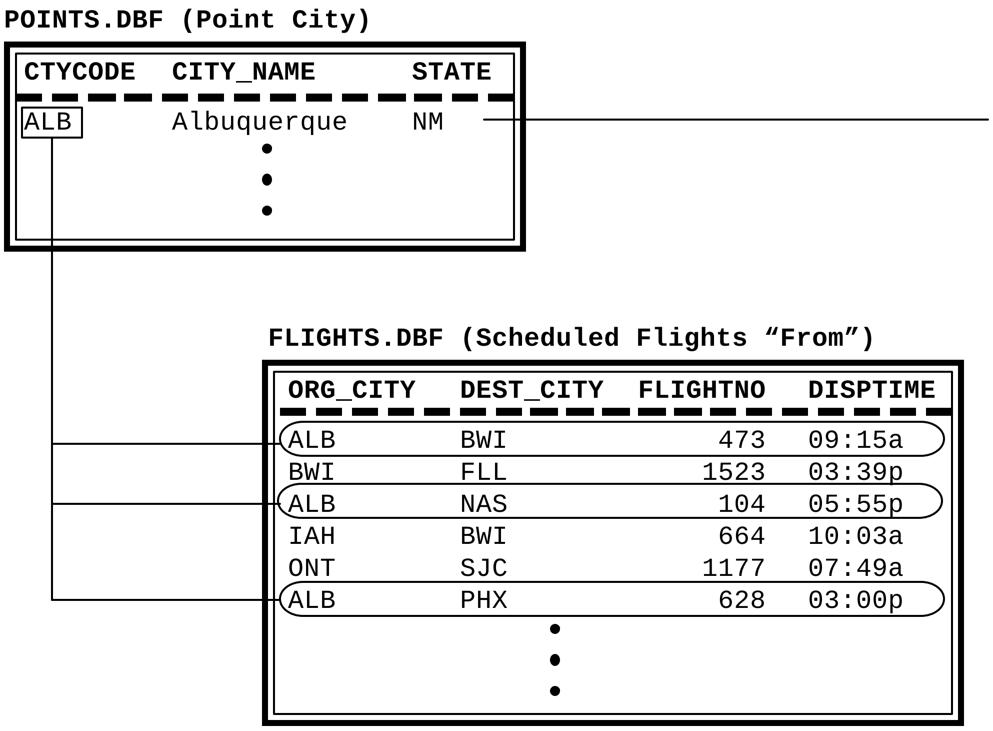
The parallel to our current Rubin family problem is direct, except we are adding gift values instead of flight listings. In order to produce a report that will list the same record more than once, we need a new composite database shape. Returning to our SWA timetable, let’s review how we would construct it as if we were doing so by hand.

First, we list each point city, sorting them alphabetically. We then select the first city (Albuquerque) and list all the flights to other cities. Figure 15-D demonstrates the procedure graphically. Next, we find all the flights from other cities to Albuquerque. (Figure 15-E.)[1]

[1] “Point city” is my term for a city served by the airline. Most airlines use the term “destination city”, but use of this term might be misconstrued, in this context, as the city to where flights arrive. For timetable purposes, we need to define the point city as both.

The flight book from SWA represents a nearly identical problem to Ruth’s. The airlines exchange planes -- and passengers -- between cities.

Figure 15-C: Page from Southwest Airlines flight timetable.



POINTS.DBF (Point City)

CTYCODE	CITY_NAME	STATE
ALB	Albuquerque	NM
	• • •	

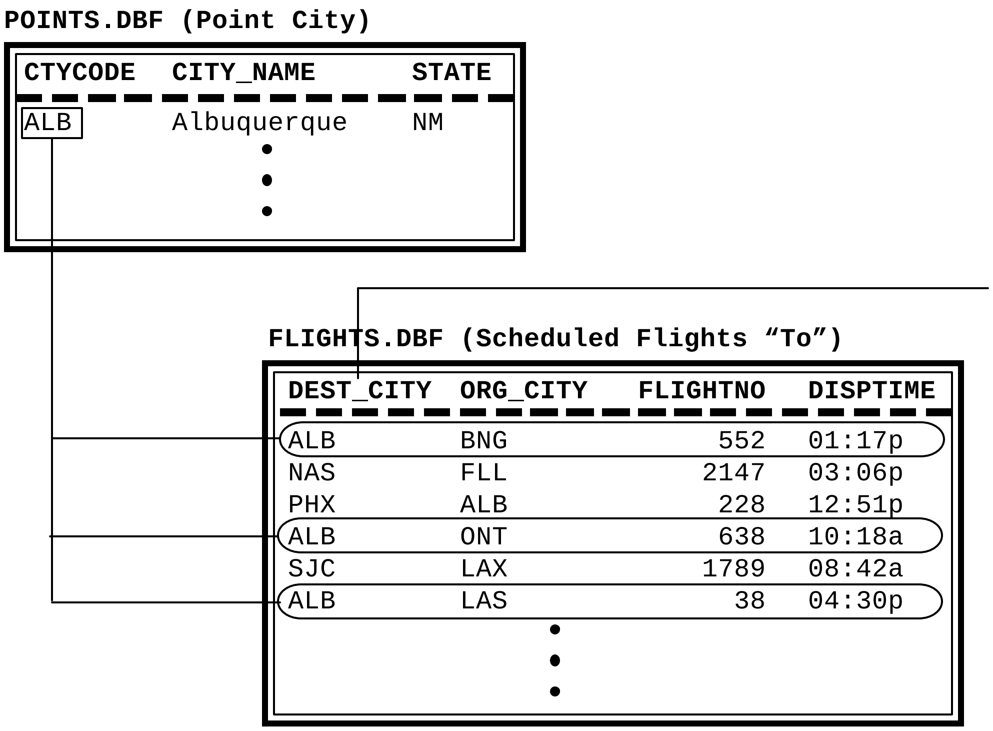
Each record in the 'point city' table is the origination city in the 'flights' table. This produces a one-to-many (scan) natural relation from 'point' to 'flight'.

FLIGHTS.DBF (Scheduled Flights "From")

ORG_CITY	DEST_CITY	FLIGHTNO	DISPTIME
ALB	BWI	473	09:15a
BWI	FLL	1523	03:39p
ALB	NAS	104	05:55p
IAH	BWI	664	10:03a
ONT	SJC	1177	07:49a
ALB	PHX	628	03:00p
	• • •		

Other important fields in the flights table are time, days of the week and seat capacity. Add fields for departure date, reservations, crew ids, plane id, passenger count, and revenue (among others), and you have a departures table.

Figure 15-D: One to many lookup of flights to other cities from Albuquerque.



POINTS.DBF (Point City)

CTYCODE	CITY_NAME	STATE
ALB	Albuquerque	NM
	• • •	

As in Figure 15-D, the natural relation between 'point' and 'flight' as the destination city is one-to-many. This link is performed on a different field in FLIGHTS.DBF than the one above.

FLIGHTS.DBF (Scheduled Flights "To")

DEST_CITY	ORG_CITY	FLIGHTNO	DISPTIME
ALB	BNG	552	01:17p
NAS	FLL	2147	03:06p
PHX	ALB	228	12:51p
ALB	ONT	638	10:18a
SJC	LAX	1789	08:42a
ALB	LAS	38	04:30p
	• • •		

Figure 15-E: One to many lookup of flights from other cities to Albuquerque.

If we combine the two sets of records into a single graphic, the diagram would look like Figure 15-F. Compare this diagram to the one on page 94 for a multi-scan composite database shape. Hmm.

We sort and group all the flights to or from each city and sort them within the to/from control, and then sort them by departure time. The sorting and grouping for the timetable is displayed in Figure 15-G. This arrangement of the information meets the timetable report's purpose: to assist travelers in locating and selecting flights to and from each point city. Note that point city is the primary sorting level. If the timetable used destination city, all of the "to" flights would be listed first.

Where Am I Going With This?

Count the number of flights to and from the city pair of Albuquerque and Baltimore. The number of listings is the sum of flights from Albuquerque to Baltimore and flights to Albuquerque from Baltimore. This makes perfect sense, of course, because each flight is a separate timetable event. Therefore, any point city flight count — that is, the

An airline timetable is a reference report, hence the sorting order is vital to achieve the user's purpose.

The grouping order supports the sorting by providing breaks by point city, paired city and departures versus arrivals.

To demonstrate this example within R&R, the **Data Samples Diskette** contains a few point city and flight records to simulate the flight schedule.

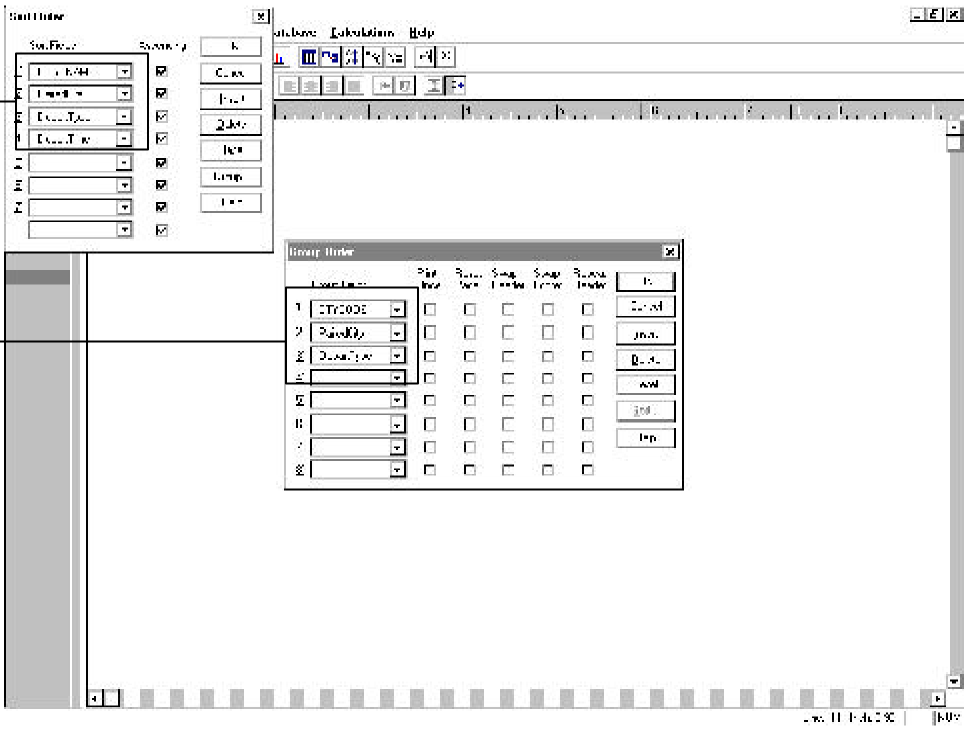
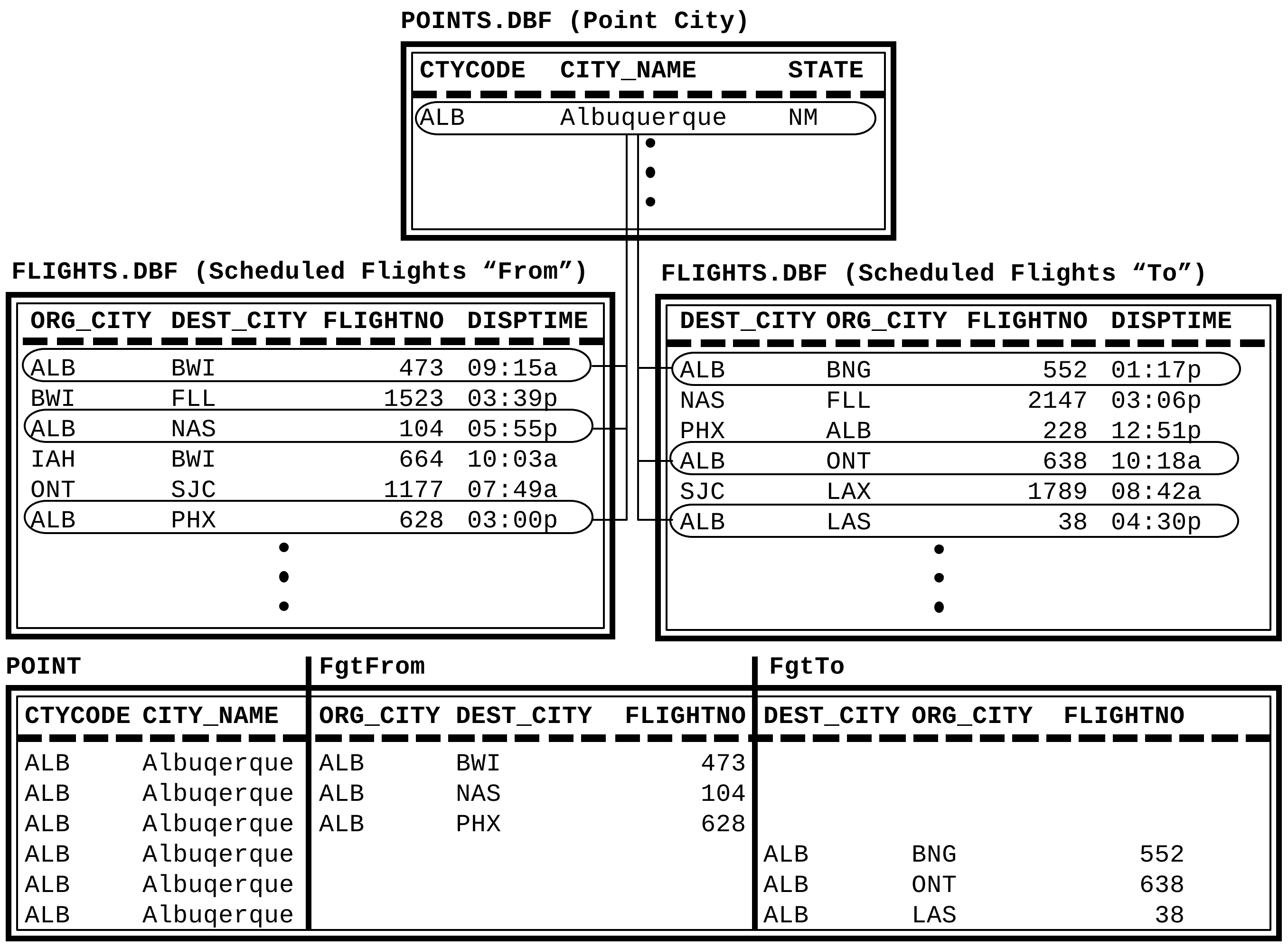


Figure 15-G: Sorting and grouping specification for airline timetable based on point city.



POINTS.DBF (Point City)

CTYCODE	CITY_NAME	STATE
ALB	Albuquerque	NM
•		
•		
•		

FLIGHTS.DBF (Scheduled Flights "From")

ORG_CITY	DEST_CITY	FLIGHTNO	DISPTIME
ALB	BWI	473	09:15a
BWI	FLL	1523	03:39p
ALB	NAS	104	05:55p
IAH	BWI	664	10:03a
ONT	SJC	1177	07:49a
ALB	PHX	628	03:00p
•			
•			
•			

FLIGHTS.DBF (Scheduled Flights "To")

DEST_CITY	ORG_CITY	FLIGHTNO	DISPTIME
ALB	BNG	552	01:17p
NAS	FLL	2147	03:06p
PHX	ALB	228	12:51p
ALB	ONT	638	10:18a
SJC	LAX	1789	08:42a
ALB	LAS	38	04:30p
•			
•			
•			

POINT		FgtFrom			FgtTo		
CTYCODE	CITY_NAME	ORG_CITY	DEST_CITY	FLIGHTNO	DEST_CITY	ORG_CITY	FLIGHTNO
ALB	Albuqerque	ALB	BWI	473			
ALB	Albuqerque	ALB	NAS	104			
ALB	Albuqerque	ALB	PHX	628			
ALB	Albuqerque				ALB	BNG	552
ALB	Albuqerque				ALB	ONT	638
ALB	Albuqerque				ALB	LAS	38

Figure 15-F: Combined diagram of one-to-many lookup from flights to and from Albuquerque.

number of timetable records — is the sum of all flights to and from the point city.[2]

Compare the above definition with the one for a multi-scan composite database shape described on page 94. They are the same because the airline timetable is based upon a multi-scan composite database shape!

* * *

The airline timetable is just one example of a common relationship between tables. A single entity — in this case a point city — has two or more one-to-many relationships with an event table. The events are

[2] A "city pair" is a flight between any two point cities.

SQuirre**L** Nuggets #2: Getting Run Over

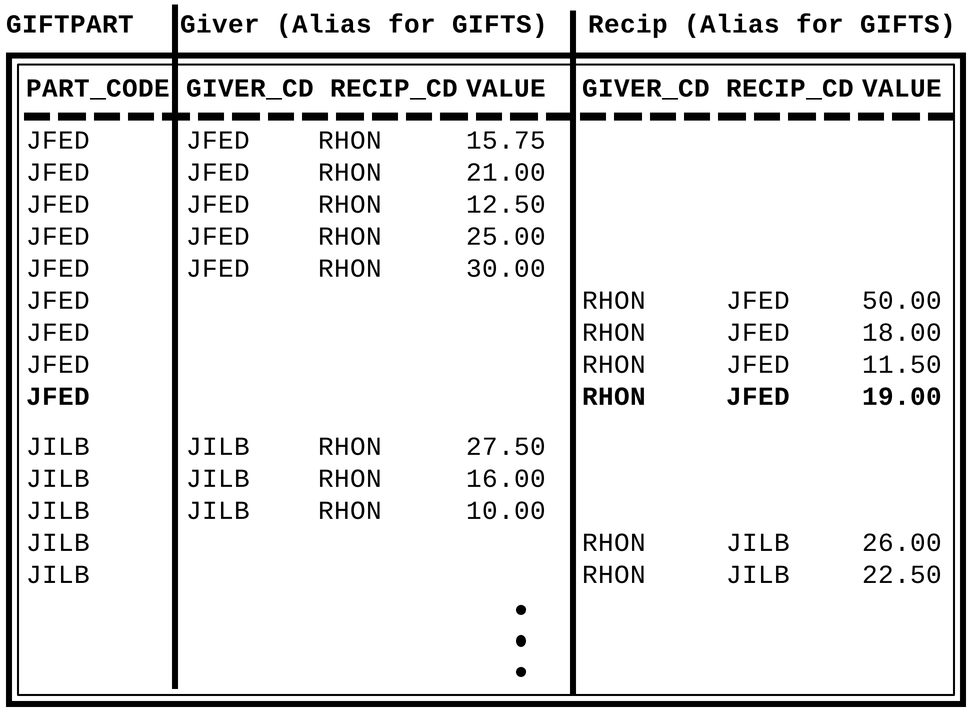
I am dumbfounded every time I think about it, but SQL versions of R&R — including Arpeggio — do not do multi-scan relationships. This restriction is not R&R's fault, however, it's SQL's. The same is true of Microsoft Access and all other programs I have reviewed.

As remarkable as it seems, the people who developed SQL did not incorporate commands to produce a multi-scan relationship. If you attempt one, it will link all of the records together via their common piece of information. If there are, say, 15 flights from Dallas to Houston and 15 return flights, SQL will not create 30 records for Dallas as the point city. It will build 225!!

The closest SQL can get to true multi-scan is a "union join", and some SQL versions I've reviewed have some adjunct commands to produce a composite database shape closer to true multi-scan. With any of these SQL versions, however, one must write a complicated SELECT statement to simulate even a poor fascimile of a multi-scan composite database.

This represents a fundamental flaw in SQL since multi-scan composite database shapes are so common and necessary. I can only attribute it to the lack of formal training in database reporting. This is no surprise since I cannot find a single resource book that even mentions database reporting in any more than a cursory way.

Replace a Rubin sibling for point city, and gift exchange partner for paired city, and you have the same multi-scan relationship as an airline timetable.



GIFTPART	Giver (Alias for GIFTS)			Recip (Alias for GIFTS)		
PART_CODE	GIVER_CD	RECIP_CD	VALUE	GIVER_CD	RECIP_CD	VALUE
JFED	JFED	RHON	15.75			
JFED	JFED	RHON	21.00			
JFED	JFED	RHON	12.50			
JFED	JFED	RHON	25.00			
JFED	JFED	RHON	30.00			
JFED				RHON	JFED	50.00
JFED				RHON	JFED	18.00
JFED				RHON	JFED	11.50
JFED				**RHON**	**JFED**	**19.00**
JILB	JILB	RHON	27.50			
JILB	JILB	RHON	16.00			
JILB	JILB	RHON	10.00			
JILB				RHON	JILB	26.00
JILB				RHON	JILB	22.50
			⋮			

Figure 15-H: Composite database diagram for *Gift Exchange Analysis* revealing multi-scan shape.