

Lesson 12

An Introduction to Database Publishing with FrameMaker

by Bradley W. Anderson

In This Chapter ...

■	An Introduction to Database Publishing	3
■	The Cost of Setting Up the Initial Database Publishing System	5
■	Publishing a Simple Directory from a Database.	9
●	Relational Database Structure	12
●	Writing SQL Queries.	15
●	Retrieving the Data from the Database	18
■	Examples of Database Publishing with FrameMaker	23
◆	Data-Driven Publishing with PatternStream™	24
●	Directory Example Using PatternStream	25
◆	Data-Driven Publishing with FrameScript	37
●	Simple Directory FrameScript Code	38
◆◆	Automated Publishing with MML.	47
●	Sample SQL Query that Returns MML Syntax.	48

Last modified on January 4, 2010

- ◆◆ Automated Publishing with XML50
- Sample SQL Query that Returns XML Syntax51
- ◆◆ Automated Publishing with MIF53
- Sample SQL Query that Returns MIF Syntax54
- Summary60
- Send Feedback61

■ An Introduction to Database Publishing

Database publishing is the process by which information is automatically taken from one or more data sources—including any number of databases, flat files, and images—and assembled into a complete document. Examples of publications produced by database publishing include directories, catalogs, price lists, invoices, financial statements, and reports. This chapter is written for those individuals beginning the process of publishing information from a database. It explores different methods of retrieving data (both static and dynamic) and publishing data with FrameMaker.



With the release of FrameMaker 7, the desktop version of FrameMaker is not licensed for entirely-automated, server based publishing. FrameMaker Server 7.0 has the same technology as the desktop version of FrameMaker but has a license for use in server-based solutions. See Adobe's website for more information about FrameMaker Server 7.0 and download their white paper on database publishing.

Automated Tagged Publishing vs. Data-Driven Publishing

Database publishing can be segregated into two broad categories—automated tagged publishing and data-driven, dynamic publishing. What's the difference? Automated tagged publishing is the process by which a flat file (basically a big text file with special markups or tags in it) is parsed and published with FrameMaker. Data-driven publishing is the process by which data is retrieved (in real time) from the database and published (in real time) into FrameMaker—there is no need for an intermediary markup language or text file. Although automated tagged publishing solutions are partially data-driven (e.g., they decide what to publish based on values in the tagged file they are parsing), the entire process is not data-driven.

There are many applications on the market that claim to be data-driven publishing solutions when, in fact, they are nothing more than sophisticated flat-file parsers. Many database publishing applications do not communicate directly with a database when they publish. Many use an intermediary markup language that the application parses when they publish. In other words, a markup file such as MML, XML, or MIF needs to be created from the database. Then the database publishing application parses the markup file and “publishes” the information. Automated publishing with FrameMaker can be done with MML, MIF, XML, Miramo, or Unimerge.

Database Publishing Applications

There are many applications on the market for database publishing. The following database publishing applications work (at some level) with FrameMaker.

Data-Driven Publishing Applications

- ☉ BookDB from Sabern—
www.sabern.com/eng/index.htm?show=bookdbxml.htm
- ☉ CatBase from CatBase Software Ltd.—*www.catbase.com*
- ☉ Catalogger from Guidance—*www.guidance.nl/en/page/catalogger-framemaker*
- ☉ **FrameScript**® from Finite Matters Ltd.—*www.framescript.com*
- ☉ **PatternStream**® from Finite Matters Ltd.—*www.patternstream.com*

Automated Tagged Publishing Applications

- ☉ **Miramo** from Datazone Ltd.—*www.miramo.com*
- ☉ Unimerge from Refined Reports—No Web site available

Variable Printing Applications

- ☉ **DL Pager** from Datalogics—
www.datalogics.com/products/pager/pager.asp

Author's Choice

The author has been involved with database publishing for many years. After evaluating the tools on the market, he chose PatternStream as his preferred database publishing solution. In addition to PatternStream, he also uses FrameScript for very simple database publishing projects (like the example used in this chapter). In 2004, the author joined the Finite Matters Ltd.'s team full-time and continues to provide automated publishing solutions with PatternStream.

The examples in this chapter will give you a representative idea of what is involved in database publishing. Even though a couple of examples refer to specific database publishing applications, this chapter is not meant to endorse any particular database publishing application or solution. There is not enough room to lay out a step-by-step example for each product.

■ The Cost of Setting Up the Initial Database Publishing System

The costs involved in a database publishing project can be hard to quantify. Many (if not most) companies fail to understand or are unable to quantify the costs involved. Many software vendors selling database publishing solutions are not clear on what additional costs are involved other than the software they are selling. The costs involved in database publishing encompass more than just the software, and will be the highest up front. Once the system is in place, the savings resulting from the now-automated process should pay for the initial setup costs very quickly (see [Table 1](#)).

Table 1 – High-Level Project Steps and Time Requirements

High-Level Project Steps	Percentage of Time
Setting up the database	30%–60%
Getting information from the database to publish	10%–30%
Setting up the publishing application	10%–40%
On-going Support	0%–20%

Setting up the Database

The first cost in database publishing is the cost of the database itself. This step is the most important to consider as it will affect the ease of publishing and the actual content of your final product. Setting up and populating the database will comprise 40%–60% of the time necessary to set up a database publishing solution. This could be the most costly part of the project as well—usually involving a database administrator. Setting up the database can be the most important step in any project. The structure of the database will determine the flexibility with which the data is retrieved or output.

Before you can set up the database, you need to answer some questions:

- ① Who will set up the database structure and populate the data?
- ② Will you use an off-the-shelf solution (such as CatBase) or will you set up a custom database solution with a database such as Microsoft Access, FileMaker Pro, SQL Server, MySQL, SyBase, DB2, or Oracle, along with possible flat files such as Microsoft Excel spreadsheets, XML or ASCII?
- ③ How many people will be involved in setting up the database and populating it?
- ④ How much time will this take?

Getting Information from the Database to Publish

Getting the data from the database and putting it in the desired order and format is the next big budget hurdle in a database publishing project. Before you start, you'll need to answer some more questions:

- ③ How will you get the data to FrameMaker for publishing?
- ③ Will you “dump” the data from the database in a markup language such as MML, XML, MIF, or some other markup language that then needs to be run through another database publishing application that understands the markup and then output in Frame?
- ③ Will the database be queried directly and the data pushed into FrameMaker on the fly?

If the data is “dumped” into a markup file, the initial cost involved is usually more because a database administrator is usually required. If the data is directly queried from the database, the costs are usually lower because the database administrator is only needed to answer occasional questions from the publishing staff person performing the queries during the initial setup. After the initial setup using either technique, the only further cost will be for executing the queries. The time requirement for getting the initial data (writing the SQL queries and/or reports) will be 10%–30% of the project.

Setting up the Publishing Application

The next step in database publishing is setting up the publishing application. This step includes designing the final FrameMaker templates, getting the data (whether directly from the database or from static markup files), and putting the data into the template. This step will comprise 10%–30% of the time required to complete the project.

Here are some questions you'll need to answer before you set up the publishing application:

- ③ Are you using a solution that will need to parse markup files generated from the database?
- ③ Are you using a solution that will query the data directly from the database?
- ③ Will you design the FrameMaker templates yourself?

Ongoing Support

The final cost to consider when starting a database publishing project is the cost of ongoing support. In order to choose your method of ongoing support, you need to answer these questions:

- ① What will future editions of your project cost?
- ② How often will you need to upgrade the software/hardware involved?
- ③ Will you need further support from a database administrator to build custom queries?
- ④ Will you need to maintain a support contract on the software or process you are using?
- ⑤ What is the cost of keeping your employee(s) current in the software used versus outsourcing the publishing?

When considering the cost of implementing a database publishing project, make sure to consider all of these costs and not just a single cost (such as the price of software). The cost of the software may only be a small portion of the total cost of the project.

Firm Directory		AMERICAN BENEFIT CORP	
<p>F.M.L. 05448743</p> <p>AMERCO INC Trade Name: UHALL RENTAL SYSTEM 1325 Antromore Way Bensenville, IL 60015-1201 Phone: (708) 488-8300 Edward J. Shoen Ch, Bu, Fr Gary W. Kneffeler Sec, GC Carol H. Herten VP James P. Shoen VP</p> <p>SC: 4236, 3531, 4111, 7359, 7513, 7519 household goods, ware- house equipment, accessories; insurance carriers; life insurance annuities; rental and leasing; tool rental; lawn and garden equipment; rent-a-car; rental, without driver/assistant rental.</p>	<p>F.M.L. 18875089</p> <p>AMERICA'S RENT-TO-OWN CENTER INC Trade Name: AMERICA'S SALES & LEASING 11 N Highland St. Chicago, IL 60620-1853 Gary D. P W. Frank Sprigler Ex VP Judith C. Cooper Sec Tr</p> <p>SC: 7359 furniture rental; electronic equipment rental, except computers; appliance rental.</p>	<p>F.M.L. 064992647</p> <p>AMERICAN ARBITRATION ASSOCIATION INC 140 W 14th St. New York, NY 10020-1200 Phone: (212) 684-4000 William K. Saxe P, CO SVP John J. Emmert Jr. CFO SVP Mark A. Appel SVP George H. Friedman SVP Robert M. Peterson SVP Richard W. Hamann SVP James W. Weger SVP Richard M. Kelly SVP Michael P. Heald Sec Richard D. DeWitt Cont</p> <p>SC: 2721, 2723, 7380, 8209; periodic; publishing; onlybooks; publishing; onlybook; and conciliation services; educational ser- vices.</p>	<p>F.M.L. 074481232</p> <p>AMERICAN BAPTIST ESTATES INC Trade Name: QUINCYWOOD 7550 N 16th St Phoenix, AZ 85020-4618 Phone: (602) 944-4455 Dr. Richard L. P David Ferguson Ex VP Rev. Robert SVP Linda D. Emmer Adm</p> <p>SC: 8051, 8052, 8061; convalescent home; facilities; nursing, with health care.</p>
<p>F.M.L. 027346404</p> <p>AMERI FACT INC 4500 Rockledge Rd Ste 220 Cleveland, OH 44131-2131 Phone: (216) 575-1888 Harry Gambill P Bruce Masak VP</p> <p>SC: 7381; detective services.</p>	<p>F.M.L. 878517147</p> <p>AMERICAID INC Trade Name: AMERICAD COMMUNITY CARE 4423 Corporation Ln Virginia Beach, VA 23462-3103 Jeffrey Mc Waters P Donna Ferrante CFO</p> <p>SC: 8011; health maintenance organization.</p>	<p>F.M.L. 18178621</p> <p>AMERICAN ASSAY LABORATORIES INC 1301 Grand Ave Sparks, NV 89431-0902 Phone: (702) 356-0606 George Burke P Dick Rosenbark Sec Tr Julie Alden Cont</p> <p>SC: 8735; testing laboratories.</p>	<p>F.M.L. 07691303</p> <p>AMERICAN BAPTIST HOMES OF THE MIDWEST 3185 Technology Dr Eden Prairie, MN 55344-3622 Phone: (952) 938-1175 Lois Bohan Ch, Bu Jack J. Froy P Polly Schrom Sec Tr Noel Dikman Actg</p> <p>SC: 8051, 8052, 8061; convalescent home; with continuous nurs- ing care; mental retardation hospital/home; for the mentally retarded; with health care; home; with health care; residential.</p>
<p>F.M.L. 798680893</p> <p>AMERI STAFFING 2115 Main St Dayton, OH 45422-2411 Phone: (513) 224-0400 Dan Kee Over</p> <p>SC: 7383; employee leasing services.</p>	<p>F.M.L. 11819697</p> <p>AMERICAN CORP 730 Spitzer Dr Lombard, IL 60148-4011 Phone: (800) 420-4355 George A. Kessler P</p> <p>SC: 7389; subcontracting services.</p>	<p>F.M.L. 04678077</p> <p>AMERICAN ASSOCIATION OF RETIRED PERSONS Trade Name: AARP 601 E St NW Washington, DC 20040-0001 Phone: (202) 434-2277 Douglas H. P William COO Clifford Cont Steven S. Zaleski GC Robert Ex Dir James Connects James P Charles W. Allen Public Constance Rich Thomas C. Secs Anne Dir Nancy Dir John Dir Jan Dir</p> <p>SC: 8300; health systems agency; social change association; social service information exchange.</p>	<p>F.M.L. 1801893630</p> <p>AMERICAN BAPTIST HOMES OF WASHINGTON INC Trade Name: JERSON PARK RETIREMENT 23600 Marine View Dr S Sea Mar, WA 98148-7352 Phone: (206) 824-4000 David B. Ferguson P George Goughlin VP Andrew Dir</p> <p>SC: 8051, 8059; skilled nursing care; facilities; convalescent homes.</p>
<p>F.M.L. 878122603</p> <p>AMERICA CENTRAL CORP 5880 NW 66th Ave Miami, FL 33142-2409 Phone: (305) 471-1587 Ruben P</p> <p>SC: 8122, 8141; air passenger carrier; schedule management ser- vices.</p>	<p>F.M.L. 188551429</p> <p>AMERICAN ALLIANCE ALWAYS AVAILABLE Trade Name: OMIN PATROL 2481 Lindberg St Auburn, CA 95602-9061 Phone: (916) 888-8114 Chester Eckert P Trigid Eckert Sec Joseph Asst Sec They Off Mgr</p> <p>SC: 7699; sewer cleaning & rodding.</p>	<p>F.M.L. 03876187</p> <p>AMERICAN AUTO SERVICE INC Trade Name: AAS 11241 Rockledge Blvd North Hollywood, CA 91601-2466 Phone: (818) 761-5220 Barry P Henry P Arthur Ex VP Joseph VP Marg VP</p> <p>SC: 7336; general automotive repair shops.</p>	<p>F.M.L. 058418567</p> <p>AMERICAN BECHTEL INC 50 Beale St San Francisco, CA 94105-1813 Phone: (415) 768-1244 Peter Ch, Bu Peter P Arthur SVP Robert SVP J.P. SVP Robert SVP V.P. SVP John SVP James SVP</p> <p>SC: 1628, 8711, 8742; industrial plant construction; power plant construction; construction & civil engineering; construction project management; consultant.</p>
<p>F.M.L. 849284526</p> <p>AMERICA K INTERNATIONAL INC 2 Main St New York, NY 10013-5003 Phone: (212) 307-5434 Stephen Kong Ch</p> <p>SC: 8142; new business start-up consultant.</p>	<p>F.M.L. 86859482</p> <p>AMERICAN ALPHA DATA CORP 108 N. Brown St Ste 201 Arlington, TX 76012-2868 Phone: (817) 266-6280 Joseph J. P</p> <p>SC: 7374; data processing service.</p>	<p>F.M.L. 05353072</p> <p>AMERICAN BANCORPORATION 102 Main St Waltham, MA 02451-2726 Phone: (617) 253-3000 Anthony Ch, Bu, CEO James P Paul P Brent P</p> <p>SC: 6022, 6162, 7374; state commercial bank; mortgage brokers; using own monies; data processing service.</p>	<p>F.M.L. 804201572</p> <p>AMERICAN BEHAVIORAL RESEARCH ASSOCIATION 1 West St New York, NY 10401-2315 Phone: (516) 473-7000 Richard CEO John CFO Richard Ex VP, GM Andrew SVP</p> <p>SC: 8741; financial management for business.</p>
<p>F.M.L. 02479060</p> <p>AMERICA WORKS OF NEW YORK INC 278 Van 14th Pl New York, NY 10018-3011 Phone: (212) 244-5627 Abraham P, Tr</p> <p>SC: 7381; employment agencies.</p>	<p>F.M.L. 80859482</p> <p>AMERICAN ANALYTICAL SERVICES INC 110 Technology Pl Norcross, GA 30092-2008 Phone: (770) 744-4200 Robert P Robert P Billy VP Denise VP</p> <p>SC: 8734; testing laboratories.</p>	<p>F.M.L. 020450880</p> <p>AMERICAN BAPTIST ASSEMBLY INC Trade Name: GREER LANE CONFERENCE CENTER W5311 State Road 218 & 4 Grant Lake, VA 20641-0642 Phone: (817) 380-1326 Neil Sec Tr Neil P</p> <p>SC: 1981, 1992; membership basis organization; hotel/publi- cist courses.</p>	<p>F.M.L. 157628094</p> <p>AMERICAN BENEFIT CORP 401 11th St Huntington, WV 25701-2218 Phone: (614) 525-0331 Matthew P, CO SVP Kenneth Ex VP E. SVP Thomas Sec Tr Barbara Cont</p> <p>SC: 8042, 8048, 8059; health resources; consulting; health-care employee programs; administrative; financial; consultant.</p>
<p>F.M.L. 793096924</p> <p>AMERICA'S FINEST SERVICE CO INC 407 Parkway Dr Greensboro, NC 27401 Phone: (910) 379-0156 Wynetta P Robert VP</p> <p>SC: 7349; cleaning service, industrial or commercial; mail deliv- eration; control or fee basis.</p>	<p>F.M.L. 182824078</p> <p>AMERICAN ANESTHESIA SERVICES INC 2965 E. 9th Ave Walla Walla, WA 99180-3858 Phone: (509) 860-3400 Al P, CO Andy CFO Sid COO</p> <p>SC: 8099; medical services organization.</p>	<p>F.M.L. 076614767</p> <p>AMERICAN BAPTIST ELDER MINISTRIES OF MASSACHUSETTS 1810 Chapman Way Hingham, MA 02044 Phone: (617) 780-0226 Rev. Ch</p> <p>SC: 8051, 8122, 8361; skilled nursing care; facilities; adult day care center; residential care.</p>	<p>F.M.L. 157628094</p> <p>AMERICAN BENEFIT CORP 401 11th St Huntington, WV 25701-2218 Phone: (614) 525-0331 Matthew P, CO SVP Kenneth Ex VP E. SVP Thomas Sec Tr Barbara Cont</p> <p>SC: 8042, 8048, 8059; health resources; consulting; health-care employee programs; administrative; financial; consultant.</p>

Figure 1 Directory page sample.



■ Publishing a Simple Directory from a Database

Figure 1 is a sample page from a large directory published from a relational database. In this example you will see the decisions that need to be made when you embark on a database publishing project. Although most database publishing projects are much more complex than this and include items such as images and tables, this examples gives a basic idea of what is involved in database publishing and the approach you should use for your project.

Determining the Output of a Directory Listing

Now take a closer look at one of the directory entries from this sample. At first glance, the entry looks fairly easy and straight-forward. We are outputting a company's FML number, name, trade name, address, city, state,

zip+5, zip+4, phone number, list of executives and his/her titles, SIC number(s), and SIC description(s). Each of these items has a particular paragraph style associated with it (see [Figure 2](#)).

AMERICAN ASSOCIATION OF RETIRED PERSONS

TRADE NAME AARP
 601 E St Nw
 Washington, DC 20049-0001
 Phone (202) 434-2277
 Norman Stengel Pr
 Richard Henry COO
 Clifford J Rose Cont
 Steven S Zaleznick GC
 Horace B Deets Ex Dir
 James R Holland Commctns
 James A Maigret Fin Plng
 Charles W Allen Publs
 Constance Swank Rsch
 Thomas C Nelson Svcs
 Wayne F Haefer Dir
 C Anne Harvey Dir
 Mary Anne Riesenbergl Dir
 Johc C Rother Dir
 Jan Stefanov Dir

SIC 8399: health systems agency; social change association; social service information exchange.

Figure 2 Sample entry from directory.

Now take a closer look at the decisions that need to be made for each entry:

- ☉ Does the company have a trade name? If so, output it. If not, don't output the default text "Trade Name" or the paragraph allocated for it.
- ☉ Does the company have a zip+4 code? If so, output the zip+5 and the zip+4 with an en dash in between them. If not, output only the zip+5 without the en dash.

- ② Does the company have a phone number? If so, format the phone number as (###) ###-#### and output it. If not, don't output the paragraph allocated for it.
- ② Does the company have any executives listed? If so, output each of them in a separate paragraph. If not, don't output the paragraph(s) allocated for them.
- ② If the company has executives, does the executive have a title? If so, output the executive's name along with a tab and the executive's title. If not, output only the executive name without the tab (so you're not left with a leader and no text following it).
- ② Does the company have any SIC codes? If so, how many? If one, output the SIC code (with a character format applied to make it red) followed by a colon. If more than one, output each SIC code (with a character format applied to make it red) with a comma in between each SIC code. The last SIC code should be followed by a colon.
- ② If the company has one or more SIC codes, how many SIC descriptions does the company have? If one, output the description followed by a period. If more than one, output each SIC description followed by a semicolon, except for the last description, which should be followed by a period.
- ② One item that is not visible to the naked eye is the index marker for the company name that gets outputted with the company name.
- ② In what order do the entries appear? Alphabetically or in a special sequence?

Now that you've looked at output and the desired layout and data, you need to look at the actual database structure to see how the data is organized and how to access it. We will now look at the database structure and introduce the structured query language (SQL) that will be used to request the data from the database.

● Relational Database Structure

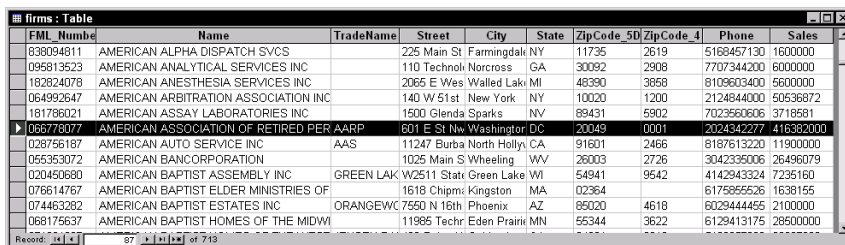
The database used for the examples shown in this chapter uses a relational database structure (some of the new document repositories use an object-oriented or XML/SGML data structure). This simply means that the data is divided and stored in separate locations in the database to keep the data from being repeated. For example, if you had a database of customers and kept track of each order that was placed by each customer, you could store the data in two manners. You could reenter the customers' contact information (address, phone number, etc.) along with each order. Or you could keep the customers' contact information in one location, the orders in another, and use a customer number to "relate" the contact information to the order information.

Database Tables

In this example, the database used to generate the directory is a simple Access database. It contains three tables—Firms, Executives, and SIC_numbers. These three tables are related to each other by the ID field called FML_Number. In other words, each record in each table will have an FML_Number—this allows the executives and SIC information to be associated with the appropriate firm.

Firms Table

The Firms table contains related company information such as the company name, trade name, street address, city, state, zip5, zip4, and phone number. Notice the FML number 066778077 for the highlighted AARP entry. This is the number used to related the entries in the next two tables (see Figure 3).



FML_Number	Name	TradeName	Street	City	State	ZipCode_5D	ZipCode_4	Phone	Sales
836094811	AMERICAN ALPHA DISPATCH SVCS		225 Main St	Farmingdale	NY	11735	2619	5168457130	1600000
095813523	AMERICAN ANALYTICAL SERVICES INC		110 Technol	Norcross	GA	30092	2908	7707344200	6000000
182824078	AMERICAN ANESTHESIA SERVICES INC		2065 E Wes	Walled Lake	MI	48390	3858	8109603400	5600000
064992647	AMERICAN ARBITRATION ASSOCIATION INC		140 W 51st	New York	NY	10020	1200	2124844000	50536872
181786021	AMERICAN ASSAY LABORATORIES INC		1500 Glenda	Sparks	NV	89431	6902	7023960606	3718581
066778077	AMERICAN ASSOCIATION OF RETIRED PER AARP	AARP	601 E St Nw	Washington	DC	20049	0001	2024342277	416382000
028756187	AMERICAN AUTO SERVICE INC	AAS	11247 Burba	North Holly	CA	91601	2466	8187613220	11900000
055353072	AMERICAN BANCORPORATION		1025 Main S	Wheeling	WV	26003	2726	3042335006	26496079
020450680	AMERICAN BAPTIST ASSEMBLY INC	GREEN LAK	W2511 State	Green Lake	WI	54941	9542	4142943324	7235160
076614767	AMERICAN BAPTIST ELDER MINISTRIES OF		1818 Chipm	Kingston	MA	02364		6175865526	1638155
074463282	AMERICAN BAPTIST ESTATES INC	ORANGEW	7550 N 16th	Phoenix	AZ	85020	4618	6029444455	2100000
068175637	AMERICAN BAPTIST HOMES OF THE MIDWI		11985 Techr	Eden Praini	MN	55344	3622	6129413175	28500000

Figure 3 Firms table in Access.

Executives Table

The Executives table contains the executive names and titles, and the sequence in which the names should appear. Notice that the highlighted executives all have the FML number 066778077 to associate these names with AARP (see Figure Figure 4).

FML Number	Seq	Exec	Title
066303462	0040	Fernando E Gutierrez	Cont
066303462	0050	Gerald Chinloy	Asst Cont
066778077	0010	Norman Stengel	Pr
066778077	0020	Richard Henry	COO
066778077	0030	Clifford J Rose	Cont
066778077	0040	Steven S Zaleznick	GC
066778077	0050	Horace B Deets	Ex Dir
066778077	0060	James R Holland	Commctns
066778077	0070	James A Maigret	Fin Plng
066778077	0080	Charles W Allen	Publs
066778077	0090	Constance Swank	Rsch
066778077	0100	Thomas C Nelson	Svcs
066778077	0110	Wayne F Haefer	Dir
066778077	0120	C Anne Harvey	Dir
066778077	0130	Mary Anne Riesenber	Dir
066778077	0140	Johc C Rother	Dir
066778077	0150	Jan Stefanov	Dir
066897745	0010	Duane Bruns	Ch Bd
066897745	0020	Elwin Nordberg	V Ch Bd
066897745	0030	Bud Morrow	Sec
066897745	0040	John A Nelson	Admr

Figure 4 Executives table in Access.

SIC_numbers Table

The final table contains the SIC numbers and descriptions for each firm. The FML number 066778077 associates the SIC number and descriptions to the AARP entry.

Order	FML_Number	Sic	descr
002558	066064742	5699	formal wear
003013	066303462	7215	laundry machine routes, coin-operate
002482	066778077	8399	health systems agency
002482	066778077	8399	social change association
002482	066778077	8399	social service information exchange
003036	066978586	8051	skilled nursing care facilities
002518	067210161	8399	fund raising organization, non-fee bas
002886	067409094	7221	photographic studios, portrait
002886	067409094	7221	transient photographer

Figure 5 SIC_numbers table in Access.

● Writing SQL Queries

Introduction to SQL

To communicate with a database, the most commonly used method is the structured query language (SQL). SQL (pronounced “sequel”) statements are issued to the database to return the desired data. A general SQL statement looks like the following:

```
SELECT <<field(s)>> FROM <<table(s)>>
```

This statement simply tells the database to select one or more fields from a particular table in the database. For example, if the following SQL statement is issued:

```
SELECT Name FROM firms
```

the database will return a list of all company names in the database. A complex SQL statement looks like this:

```
SELECT <<DISTINCT>> <<field(s)>> FROM <<table(s)>>  
WHERE (<<some value>> <<(= <> <)>> <<some value>>)  
ORDERBY <<field(s)>>  
GROUPBY <<field(s)>>
```

The added options in the SQL statement allow you to further limit, order, and group the returned data.

Nested Queries

When you use a data-driven publishing application, queries can be “nested.” This means that a second query can be run “inside” the first query to return information based on the first query. The example shown here uses a total of four queries to produce the final directory. The first query returns the company information. The second and third queries are run as subqueries to the first—they are nested in the first query. The second query returns the executive information for the company. The third query returns the SIC numbers for the company. The fourth query is a sub-subquery that is nested under the third query. The fourth query returns the SIC descriptions for each SIC number for each company.

Queries Used in the Example

Firm Query

The Firm query returns one record for each company in the Firms table, along with the company name, phone number, street address, FML number, trade name, 5-digit zip code, 4-digit PO box number, state, and city:

```
SELECT Name, Phone, Street, FML_Number, TradeName,
ZipCode_5Digit, ZipCode_4Digit, State, City
FROM firms
```

This query returns a data value for each field named after the select statement from the table named Firms in the database. It will return each of the values for every company in the Firms table.

Executive Query

The Executive query is run as a subquery to the Firm query. It returns every executive with his/her title from the Executives table. The executives are limited to the previously selected company in the Firm query by the FML number from the firm query. The records are ordered by the values in the seq column.

```
SELECT exec, title
FROM executives
WHERE FML_Number = :FML Number
ORDER BY seq
```

This query returns an executive name and his/her title from the table named Executives for the particular company (matching the FML_Number) returned in the Firm query.

SIC Query

The SIC query is run as a subquery to the Firm query. It returns every SIC number limited to the previously selected company in the Firm query based on the FML number from the Firm query:

```
SELECT DISTINCT sic
FROM sic_numbers
WHERE FML_Number = :FML Number
```

This query returns all SIC numbers from the table named SIC_numbers for the particular company (matching the FML_Number) returned in the Firm query.

SIC Description Query

The SIC Description query is run as a subquery to the SIC query. It returns every SIC description for each SIC number. It is limited by the FML number returned in the Firm query and the SIC number returned in the SIC query.

```
SELECT descr
```

```
FROM sic_numbers  
WHERE (FML_Number = :FML Number and SIC = :SIC Number)
```

This query returns the SIC description from the table named SIC_numbers for each SIC number returned in the SIC query.

Now that you've constructed the queries needed to retrieve the desired data, you need to execute the queries to output the data.

● Retrieving the Data from the Database

Retrieving the data from the database can be done in two different manners. The SQL queries can be executed from within the database application and the data can be “dumped” into an ASCII (text) file, or the queries can be executed from an external program via open database connectivity (ODBC).

Dumping the Data from the Database

Executing the SQL queries from within the database will result in the data and any markup being written to a file on the hard drive. This file is then post-processed during publishing. This data dump becomes a “snapshot” of the database at the time the queries were executed.

Connecting to the Database via ODBC

ODBC is a commonly used protocol for connecting to a database. When you use ODBC, a database is made available to the operating system. Any ODBC-compliant application can then retrieve information from the database. FrameMaker is not natively ODBC compliant—in other words, it does not have a mechanism to connect to and communicate via ODBC. A third-party product is necessary to query information from a database into FrameMaker, or the data must be generated by the database for importing into FrameMaker. Most database publishing applications on the Mac and in Windows support ODBC database connectivity and can be used with FrameMaker for database publishing. To set up an ODBC connection, see [Setting Up an ODBC Database Connection](#).

The external application can then communicate with the database and execute the SQL queries to retrieve the data. The data can then be outputted to a file on the hard drive or sent directly into FrameMaker for publishing.

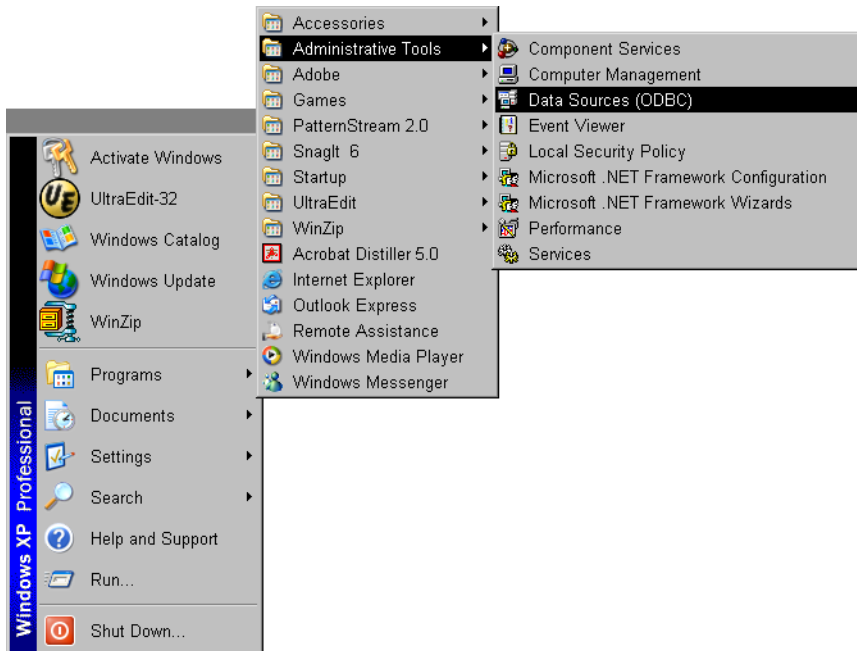
Setting Up an ODBC Database Connection

In the examples on the following pages, you will connect to an Access database via ODBC. To make this database available, you need to make the database available to the operating system via ODBC.



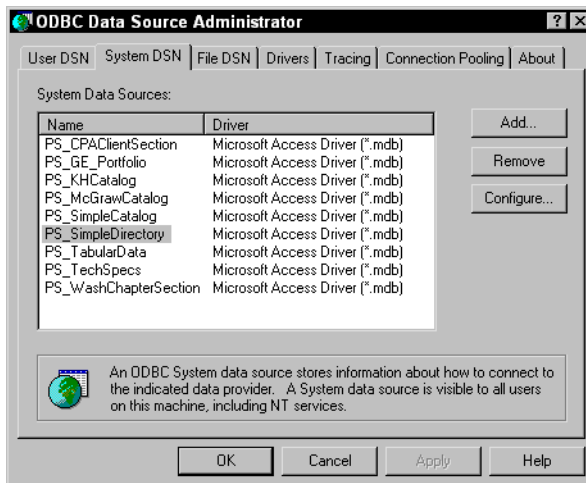
You can have this database and ODBC connection automatically setup and installed if you download and install the demo version of PatternStream available at www.patternstream.com.

1. In Windows 2000, XP, Vista, or 7 select **Start>Programs>Administrative Tools>Data Sources (ODBC)**. (In Windows 95 or 98, the ODBC data sources can be found under **Start>Settings>Control Panels**.)



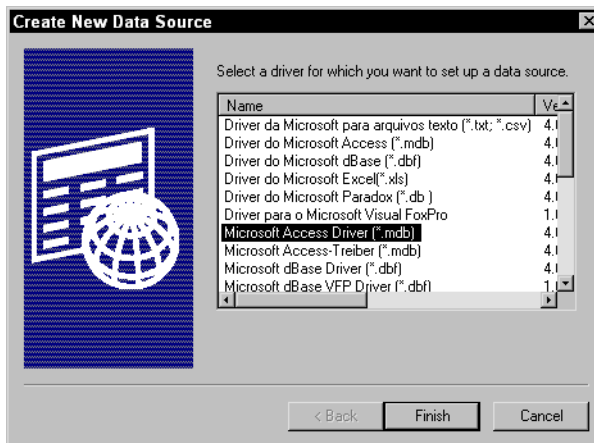
2. Click on the **System DSN** tab.

3. Click Add....



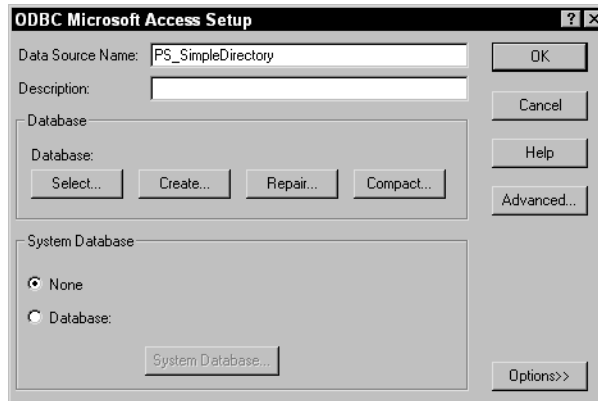
4. Select Microsoft Access Driver (*.mdb).

5. Click Finish.



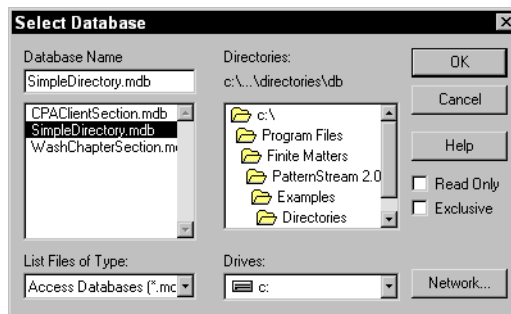
6. Enter PS_SimpleDirectory under Data Source Name.

7. Click the **Select...** button.

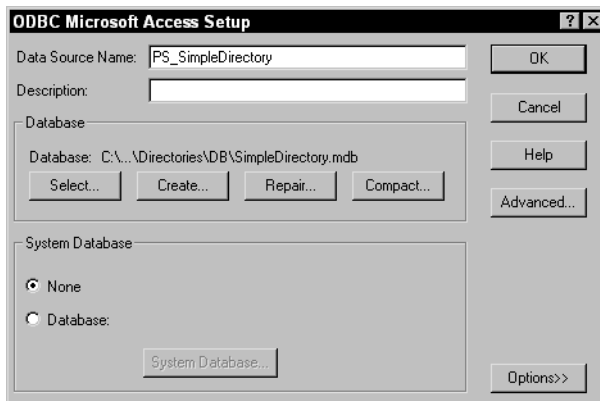


8. Navigate to the Access database entitled **SimpleDirectory.mdb**.

9. Click **OK**.



10. Click OK.



11. Click OK.

The SimpleDirectory database is now available via ODBC. A connection is made using the data source name—in this case PS_SimpleDirectory—and a username and password. The default username for an Access database is **admin** and there is not a password.

■ Examples of Database Publishing with FrameMaker

The examples in this section of database publishing project will explore five different methods of database publishing with FrameMaker. First you will look at database publishing with PatternStream—a tool that allows you to set up the entire publishing system with a graphical user interface. Next, you will set up the same project using FrameScript. The last three examples show how database publishing is done with markup languages; you'll see how to publish using MML, XML, and MIF.

◆ Data-Driven Publishing with PatternStream™

PatternStream is a plug-in to FrameMaker that communicates directly with one or more databases (either natively or via an ODBC connection) and/or flat files (such as Excel, HTML, Word, or FrameMaker documents) to retrieve data, insert the data into a FrameMaker document, and format the data on the fly. No intermediary markup files are necessary for publishing. Prototyping is a very rapid process because changes can be made and samples can be run interactively as the data is outputted directly to FrameMaker.

PatternStream's graphical user interface (GUI) allows even a newcomer to database publishing to develop with ease. PatternStream is a great tool for rapid prototyping and development for any database publishing project. The instructions provided in this book highlight the ease of setting up a database publishing project from a GUI.

● Directory Example Using PatternStream

Introduction

The following is a fairly complete step-by-step guide to setting up the directory example used in this chapter in PatternStream.

Data Link

The Data Link folder of the PatternStream interface allows you to set up one or more connections to a database, construct the queries to retrieve the data, and create variables to store the retrieved data.

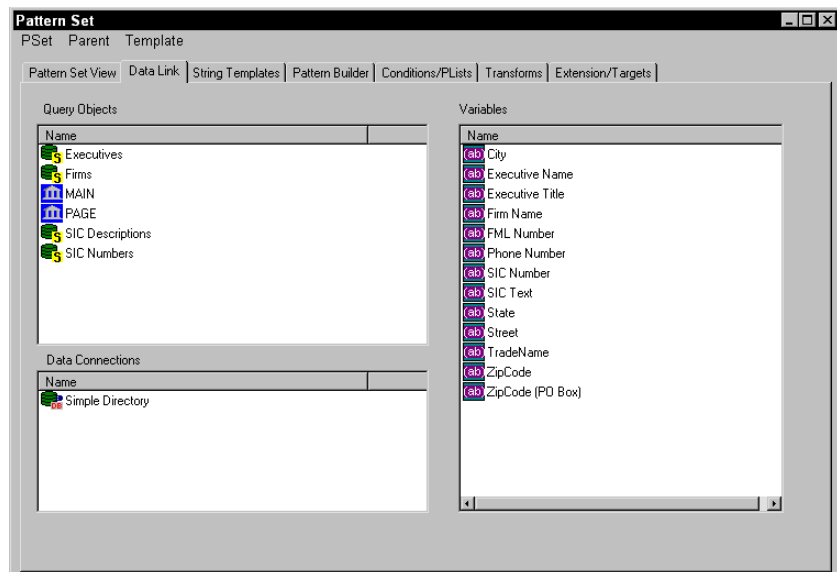


Figure 6 PatternStream's Data Link folder.

Building a Database Connection

To build a connection to the database, do the following:

1. Right-click in the Data Connections window and choose **New Object**.
2. Select **ODBC** from the Type drop-down list.
3. Name the data connection **Simple Directory**.
4. Click **OK**.

5. Type in the ODBC data source name that was created in the “Connecting to the Database via ODBC” section in Chapter 14.
6. Enter the database user ID—the user ID in Access is always **admin** by default.
7. Enter the database password—the database password in Access is always blank by default.

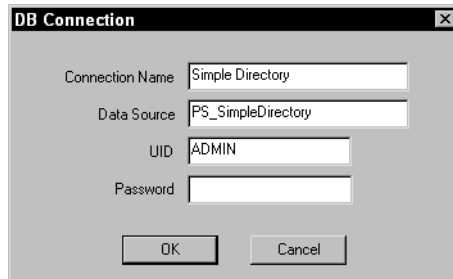


Figure 7 DB Connection dialog box.

8. Click **OK**.

Creating Variables

Next you need to set up a string variable for each piece of data retrieved from the database. To create a variable, do the following:

1. Right-click in the Variables window and choose **New Object**.
2. Name the variable and select **String** from the Type drop-down list.
3. Click **OK**.

Querying Objects

PatternStream allows you to interactively create SQL queries. To create an SQL query, do the following:

1. Right-click in the Query Objects window and select **New Object**.
2. Name the query **Firms** and choose **Select** from the Query Type drop-down list.
3. Click **OK**.
4. Select **Simple Directory** from the Connection Object drop-down list.

5. Click on the **From** tab.
6. Enter **Firms** in the Table Name field.
7. Click the **Add Table** button.

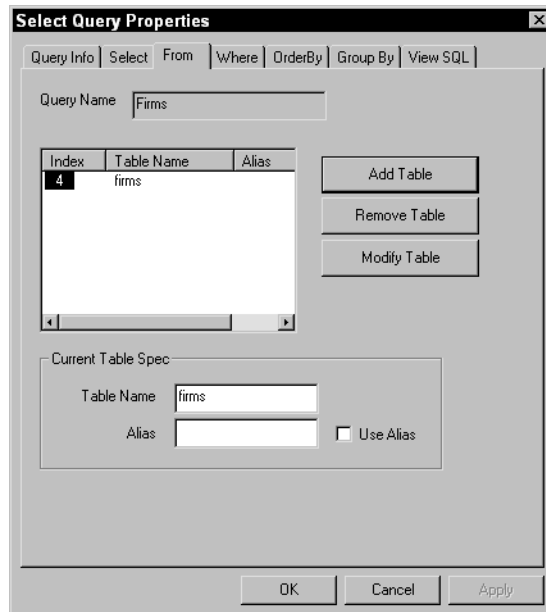


Figure 8 From tab in Select Query Properties dialog box.

8. Click on the **Select** tab.
9. Select a **PatternStream variable** from the Variable Name drop-down list.
10. Enter the **database column name** in the Column Name field.
11. Select the **Firms** table from the Table Name drop-down list.
12. Click the **Insert** button.

- Repeat Steps 9–12 for each variable and data point retrieved in this query.

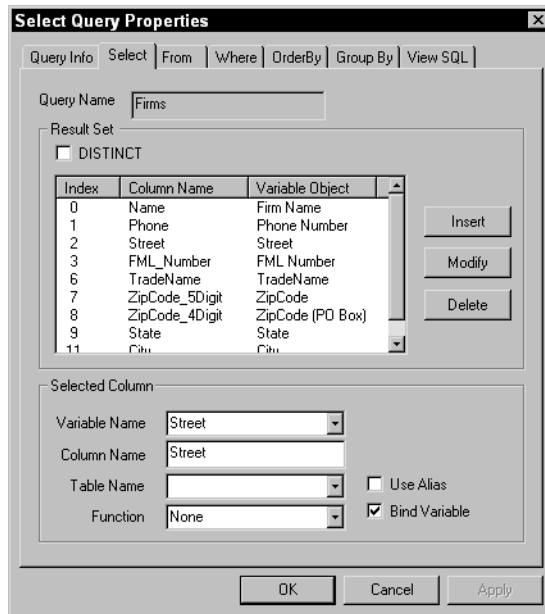


Figure 9 Select tab in Select Query Properties dialog box.

- Click **Apply**.

14. Click on the **View SQL** tab to view your query.

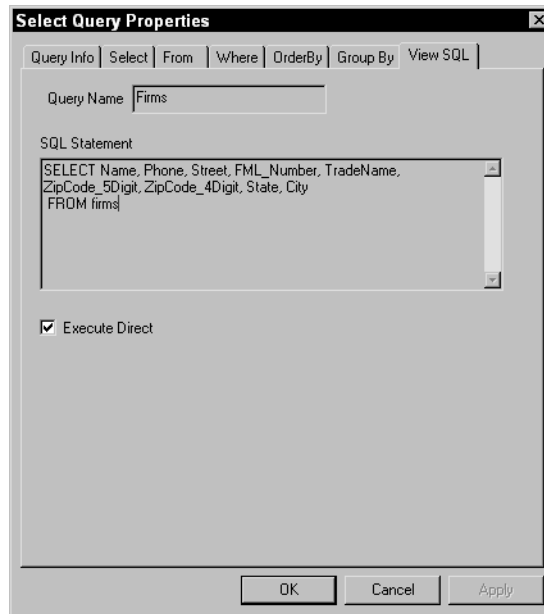


Figure 10 View SQL tab in Select Query Properties dialog box.

15. Click **OK**.

Creating String Templates

String templates are objects that allow you to publish the values stored in the PatternStream variables. In Figure B-6, notice that the City string template is made up of four variable string segments (City, State, ZipCode, and ZipCode [PO Box]) and three constant string segments (a comma and a space, two spaces, and an en dash).

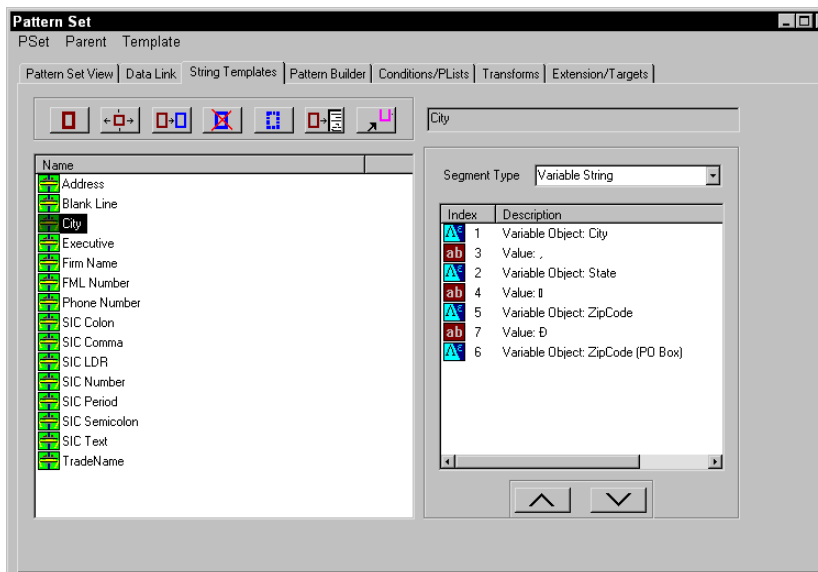


Figure 11 String templates for City and its associated variables.

To create a string segment, do the following:

1. Select **Parent>New Object** under the String Templates tab.
2. Name the string template and click **OK**.
3. To add segments to the newly created string template, choose the **segment type** from the Segment Type drop-down list.
4. **Right-click** in the Segment Type window and choose **Insert**.

If the segment type is a variable string, do the following:

- a. **Double-click** on the variable string's index number.
- b. Click the **Select Variable** button.
- c. Select the **PatternStream variable** you wish to use.

d. Click **OK**.

If the segment type is a constant string, do the following:

e. **Double-click** on the constant string's index number.

f. Type the **static text** in the String Value field, or choose the special character (such as an em dash, soft return, etc.) from the Special Character drop-down list and click the **Insert Hex Code** button.

g. Click **OK**.

5. Repeat Steps 1–5 until all the necessary string templates are created.

Building Patterns

Now that you have created a data connection to the database, have created queries to select the data values from the database, have created variables to store the data, and have created string templates to output the variable values, you are ready to create a pattern to attach a query and paragraph targets to output to FrameMaker.

To create a new pattern, do the following:

1. Click on the **PatternBuilder** tab.
2. Select **Parent>New Object**.
3. Name the pattern **Firms** and click **OK**.
4. Select the new pattern.
5. Select **Parent>Object Info**.
6. Click the **Select Query** button.
7. Select the query **Firms**.

8. Click OK.

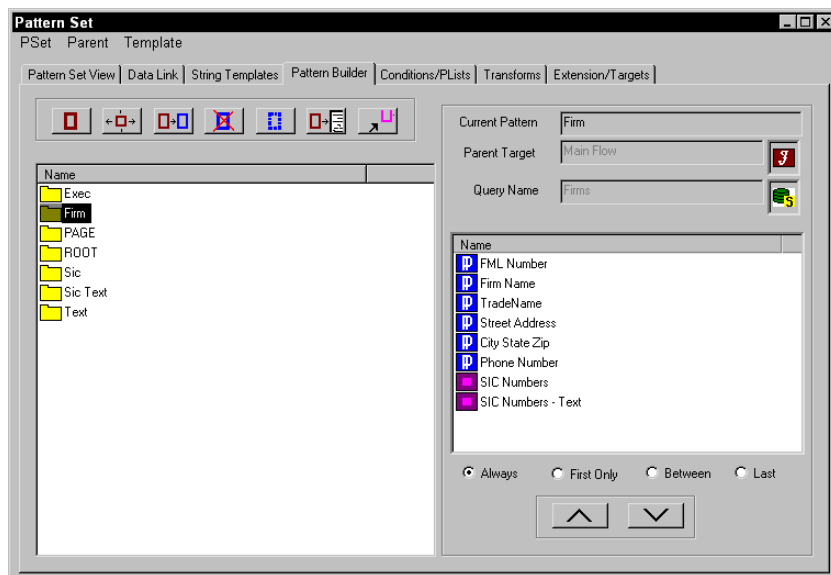


Figure 12 Creating a pattern.

Creating Paragraph Targets

Now that you have created a pattern, you need to create a target to output one or more of our string template values. For the Firm query and pattern, we need a paragraph target for FML_Number, Firm Name, TradeName, Street Address, City/State/Zip, and Phone Number.

To create a paragraph target, follow these steps:

1. **Right-click** in the right-hand window of the **PatternBuilder** tab and choose **Create Target**.
2. Name the paragraph target in the Name field.
3. Select **Paragraph** from the Target drop-down list.
4. Click **OK**. The Paragraph Target Properties window will open.
5. Click on the **String Templates** tab.
6. Click the **Attach** button.

7. Select the string template you want outputted in this paragraph.
8. Click **Apply**.

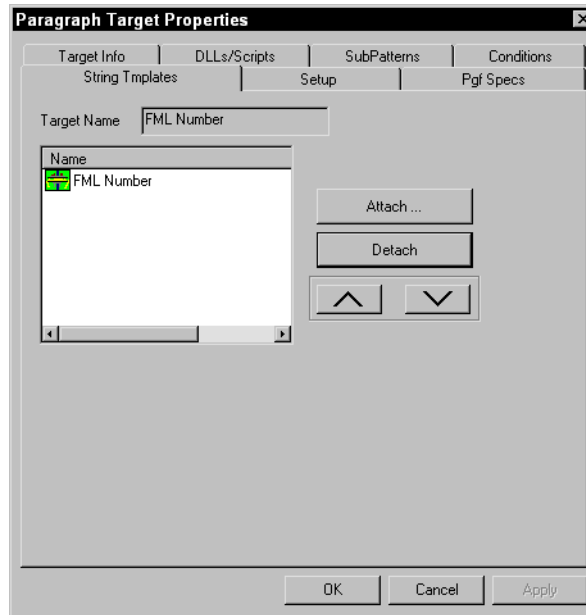


Figure 13 String Templates tab in Paragraph Target Properties dialog box.

9. Click on the **Setup** tab.
10. Make sure the **Begin New Paragraph** box is checked.
11. Select from the **Paragraph Format** drop-down list the paragraph tag in FrameMaker that you wish to use.
12. Make sure the **Generate Always** box is checked.

13. Click OK.



Figure 14 Setup tab in Paragraph Target Properties dialog box.

Creating Conditions

A condition in PatternStream allows items to be output only if a certain condition or conditions are true or false. Earlier in this example you learned about the need to not output an en dash if a particular company did not have a four-digit zip code.

To create a condition, follow these steps:

1. Click on the **Conditions/PLists** tab.
2. **Right-click** in the left-hand window and select **New Object**.
3. Name the condition **NoZip4**.
4. Select **Expression** from the Type drop-down list.
5. Under Left Hand Side, make sure **Use Literal Value** is **unchecked**.
6. Select **ZipCode (PO Box)** from the Left Hand Side Variable Name drop-down list.

7. Select **NOT EQUAL** from the Logical Operation drop-down list.
8. Make sure **Use Literal Value** is **checked** under Right Hand Side.
9. Click the **Set Value** button.
10. Select **String** and make sure the string field is blank/empty.
11. Click **OK**.

This condition can now be applied in the City string template on both the **en dash** and **ZipCode (PO Box)** segments.

PSet View

Once you have created all of the objects needed in this example, they are attached to each other to finish this project in PatternStream. The PSet View tab shows a visual representation of the entire project. This view allows you to see the logical order in which objects will be processed (see Figure 15). To output the directory from this point, just select **PSET>Run** in PatternStream.

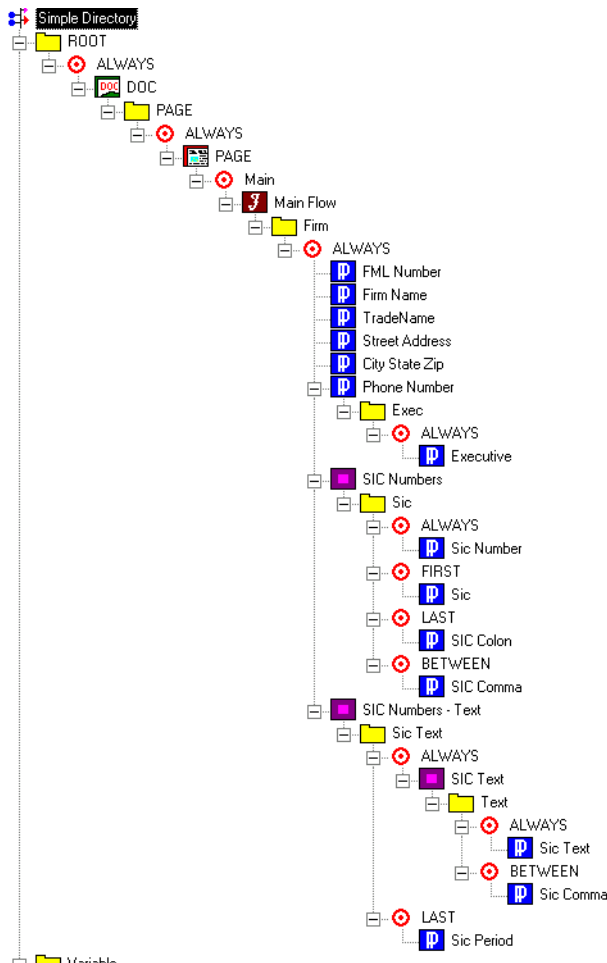


Figure 15 PSet view.

12

◆ Data-Driven Publishing with FrameScript

FrameScript is a scripting language for FrameMaker. It has support for ODBC data connections and can be used for simple database publishing projects. Very simple database publishing/mail merge applications can be done effectively with FrameScript. FrameScript also communicates directly with any ODBC-compliant database and passes the data received directly into FrameMaker, where it is formatted on the fly. Although any size of project can theoretically be done with FrameScript, the cost in development time or contracting time would soon make a large project cost prohibitive, and a solution like PatternStream would become a better option.

● Simple Directory FrameScript Code

The following FrameScript code will output the sample directory using an existing FrameMaker template named *SimpleDirectory.fm*. The code is well commented so you can see what each routine is doing. The logic is very similar to the PatternStream example you looked at earlier.

```
// -----
// -- User Defined Database Parameters
// -- Change to the correct values for your database
// -----
/*ODBC Connection Name */
SET vDSNName = 'PS_SimpleDirectory';

/*Database User ID */
SET vUserID = 'admin';

/*Database Password */
SET vPass = '';

/*Name of FrameMaker template */
SET v_Template = 'SimpleDirectory.fm';

// -----
// -- DO NOT EDIT BELOW THIS LINE
// -----

/*Gets the path to this script and puts it
in a variable called v_ScriptFolderPath */
Run GetScriptPath;

/*Opens a new document based on the variable
v_Template in the directory v_ScriptFolderPath
This script and the template must be in
the same directory to function */
Run GenerateDocument;

/*Sets and opens the datasource from which
the data will be extracted and published */
Run SetDataSource;
```

```

// -----
/*Executes the first query that returns the FML Number, Name,
Trade Name, Street Address, City, State, Zip+5, Zip+4, and Phone
from the database and outputs them to FrameMaker fully formatted
with paragraph and character format applied */
// -----
    vQuery1 = 'SELECT Name, Phone, Street, FML_Number, TradeName,
ZipCode_5Digit, ZipCode_4Digit, State, City FROM firms WHERE Name
LIKE ' +QUOTE+'amer%' +QUOTE+'';

/*Set a new EQuery object */
    New EQuery NewVar(vEQuery);
/*Set a new datasource for the EQuery object */
    Set vEQuery.EDB = vDataSource;
/*Set the query to be used for the EQuery object */
    Set vEQuery.Select = vQuery1;

/*Executes the query and returns a message if no data is returned
*/
    Run vEQuery.RunQuery;
    If vEQuery.NumCols = 0
        Write Console 'No data was returned.    ';
    EndIf

/*Loop through each record returned, format the data and output
it to FrameMaker */
    Loop While(vEQuery.AtEof = 0) LoopVar(vRecordNo) Init(1)
Incr(1)
/*Get FML Number from database and put 'F-M-L ' in front of it
and output using the 'FML Number' paragraph tag */
    Set vFML_Number = vEQuery.Fields.FML_Number;
    Set vFML_Number_Output = 'F-M-L '+vFML_Number;
    RUN Generate_Output v_Value(vFML_Number_Output)
v_Format_To_Use('FML Number');
/*Get company name from database and output using 'Firm Name'
paragraph tag */
    Set vName = vEQuery.Fields.Name;
    RUN Generate_Output v_Value(vName)
v_Format_To_Use('Firm Name');
/*Insert company name index marker */
    Get Object Type(MarkerType) Name('Index')
NewVar(vMarkerType);
    Set vIndexText = vName;
    SET vInsertionPoint = vCurrPgF;
    New Marker NewVar(vMarker) MarkerText(vIndexText)
        MarkerTypeId(vMarkerType)
TextLoc(vInsertionPoint);

```

```

/*Get Trade Name from database and put a 'Trade Name ' in front
of the trade name and output it--if it is not blank--using the
'TradeName' paragraph tag */
    Set vTradeName = vEQuery.Fields.TradeName;
    Set vTradeName_Output = 'Trade Name ' + vTradeName;
    If vTradeName <> ''
        RUN Generate_Output v_Value(vTradeName_Output)
v_Format_To_Use('TradeName');
    EndIf
/*Get the street address and output using 'Firm Info' paragraph
tag */
    Set vStreet = vEQuery.Fields.Street;
    RUN Generate_Output v_Value(vStreet)
v_Format_To_Use('Firm Info');
/*Get the city, state, zip5, zip4. Append values together and
format as a single string. Only output the hyphen and zip4 if
zip4 is not null. Output with 'Firm Info' paragraph tag. */
    Set vCity = vEQuery.Fields.City;
    Set vState = vEQuery.Fields.State;
    Set vZipCode_5Digit = vEQuery.Fields.ZipCode_5Digit;
    Set vZipCode_4Digit = vEQuery.Fields.ZipCode_4Digit;
    Set vCityStateZip = vCity + ', ' + vState + ' ' +
vZipCode_5Digit
    If vZipCode_4Digit <> ''
        Set vCityStateZip = vCityStateZip + '-' +
vZipCode_4Digit;
    EndIf
    RUN Generate_Output v_Value(vCityStateZip)
v_Format_To_Use('Firm Info');
/*Get the phone number, format it (###) ###-#### and output with
the 'Firm Info' paragraph tag. */
    Set vPhone = vEQuery.Fields.Phone;
    Run ParsePhoneNum vRawNum(vEQuery.Fields.Phone)
Returns vFormattedNum(vPhone);
    Set vPhone_Output = 'Phone ' + vPhone;
    RUN Generate_Output v_Value(vPhone_Output)
v_Format_To_Use('Firm Info');
/*Get the next record in the database. */
    vEQuery.GetNext;

// -----
/*Executes the second query that is nested within the first query
so that the executives and his/her titles can be output for each
company. This query returns the executive name and title from
the database and outputs them to FrameMaker fully formatted with
paragraph and character formats applied */
// -----
    vQuery2 = 'SELECT exec, title FROM executives WHERE
FML_Number = '+QUOTE+vFML_Number+QUOTE+' ORDER BY seq';

```

```

/* See the first query for detailed comments */
New EDB DataSource(vDSNName) NewVar(vDataSource2)
User(vUserID) Password(vPass);
Run vDataSource2.Open;
New EQuery NewVar(vEQuery2);
Set vEQuery2.EDB = vDataSource2;
Set vEQuery2.Select = vQuery2;

Run vEQuery2.RunQuery;
If vEQuery2.NumCols = 0
    Write Console 'No data was returned.    ';
EndIf

Loop While(vEQuery2.AtEof = 0)
/*Get the executive name and title and output using the 'Exec'
paragraph tag. Only output a tab character at title if the title
exists. */
    Set vExec = vEQuery2.Fields.exec;
    Set vExecTitle = vEQuery2.Fields.title;
    If vExecTitle <> ''
        Set vExec_Title = vExec + CHARTAB + vExecTitle;
        RUN Generate_Output v_Value(vExec_Title)
v_Format_To_Use('Exec');
    EndIf
        vEQuery2.GetNext;
    EndLoop
/* Close the second query and datasource. Delete the objects for
each */
Run vEQuery2.Close;
Delete Object(vEQuery2);
Run vDataSource2.Close;
Delete Object(vDataSource2);

// -----
/*Executes the third query that is nested within the first query
so that the SIC numbers can be output for each company. This
query returns the SIC number(s) from the database and outputs
them to FrameMaker fully formatted with paragraph and character
formats applied */
// -----
vQuery3 = 'SELECT DISTINCT sic FROM sic_numbers WHERE
FML_Number = '+QUOTE+vFML_Number+QUOTE;

New EDB DataSource(vDSNName) NewVar(vDataSource3)
User(vUserID) Password(vPass);
Run vDataSource3.Open;

New EQuery NewVar(vEQuery3);
Set vEQuery3.EDB = vDataSource3;
Set vEQuery3.Select = vQuery3;

```

```

Run vEQuery3.RunQuery;
If vEQuery3.NumCols = 0
    Write Console 'No data was returned.    ';
EndIf

Loop While(vEQuery3.AtEof = 0) LoopVar(vRecordNo3) Init(1)
Incr(1)
/*Get the SIC number(s) from the database. Store the first
number with a 'SIC ' in front of it. Append each subsequent
number(s) and place a ', ' between each. Store the final value
for output in the fourth query. */
Set vSic = vEQuery3.Fields.sic;
If vSic <> ''
    If vRecordNo3 =1
        Set vSic_Output = 'SIC ' + vSic;
    Else
        Set vSic_Output = vSic_Output + ', ' + vSic;
    EndIf
EndIf
vEQuery3.GetNext;

// -----
/*Executes the fourth query that is nested within the first query
so that the SIC description(s) can be output for each company.
This query returns the SIC description(s) from the database and
outputs them to FrameMaker fully formatted with paragraph and
character formats applied */
// -----
vQuery4 = 'SELECT descr FROM sic_numbers WHERE ( FML_Number
= '+QUOTE+vFML_Number+QUOTE+' and SIC = '+QUOTE+vSIC+QUOTE+' )';

/* See the first query for detailed comments */
New EDB DataSource(vDSNName) NewVar(vDataSource4)
User(vUserID) Password(vPass);
Run vDataSource4.Open;
New EQuery NewVar(vEQuery4);
Set vEQuery4.EDB = vDataSource4;
Set vEQuery4.Select = vQuery4;
Run vEQuery4.RunQuery;
If vEQuery4.NumCols = 0
    Write Console 'No data was returned.    ';
EndIf

Loop While(vEQuery4.AtEof = 0) LoopVar(vRecordNo4) Init(1)
Incr(1)

```

```

/*Get the SIC description(s) from the database. Store the first
description and append each subsequent number(s) and place a ';'
between each. Place a '.' at the end of the string. Store for
output. */
Set vDescr = vEQuery4.Fields.descr;
  If vDescr_Output <> ''
    If vRecordNo4 = 1
      Set vDescr_Output = vDescr;
    Else
      Set vDescr_Output = vDescr_Output + ';' +
vDescr;
    EndIf
  EndIf
  vEQuery4.GetNext;
EndLoop
  If vDescr_Final = '0'
    Set vDescr_Final = '';
  EndIf
  If vDescr_Final = ''
    Set vDescr_Final = vDescr_Output;
  Else
    Set vDescr_Final = vDescr_Final + ';' +
vDescr_Output;
  EndIf
/*End Query 4 */
// Close and delete the data source.
Run vEQuery4.Close;
Delete Object(vEQuery4);
// Close and delete the data source.
Run vDataSource4.Close;
Delete Object(vDataSource4);

/*End Query 3 */
EndLoop
// Close and delete the query object.
Run vEQuery3.Close;
Delete Object(vEQuery3);
// Close and delete the data source.
Run vDataSource3.Close;
Delete Object(vDataSource3);
/*Prepend the SIC variable to the description variable and output
using 'Firm Info' paragraph tag. */
Set vSic_Descr_Output = vSic_Output + ': ' + vDescr_Final +
'.';
  RUN Generate_Output v_Value('') v_Format_To_Use('Firm
Info');
  RUN Generate_Output v_Value(vSic_Descr_Output)
v_Format_To_Use('Firm Info');
Set vSic_Output = '';
Set vDescr_Final = '';

```

```

/*Insert company name header/footer $1 marker */
    Get Object Type(MarkerType) Name('Header/Footer $1')
NewVar(vMarkerType);
    Set vMarkerText = vName;
    SET vInsertionPoint = vCurrPgf;
    New Marker NewVar(vMarker) MarkerText(vMarkerText)
        MarkerTypeId(vMarkerType) TextLoc(vInsertionPoint);

/* End Query 2 */
    EndLoop
// Close and delete the query object.
    Run vEQuery.Close;
    Delete Object(vEQuery);
// Close and delete the data source.
    Run vDataSource.Close;
    Delete Object(vDataSource);

// -----
SUB GetScriptPath
    FIND DIRSEP InString(ThisScript) Back ReturnPos(ColonPos)
ReturnStatus(ColonFound);
    IF ColonFound
        GET String FromString(ThisScript) EndPos(ColonPos)
NewVar(v_ScriptFolderPath);
        ELSE
            SET v_ScriptFolderPath = ThisScript;
        ENDIF
ENDSUB
// -----

// -----
SUB GenerateDocument
    NEW Document Template (v_ScriptFolderPath+v_Template)
NewVar(v_NewDoc);
    SET currPgf = v_NewDoc.MainFlowInDoc;
ENDSUB
// -----

// -----
SUB SetDataSource
    Set ErrorCode = 0;
    New EDB DataSource(vDSNName) NewVar(vDataSource)
User(vUserID) Password(vPass);
    If ErrorCode not= 0
        MsgBox 'Error on DB create ('+ErrorCode+') Msg-
'+ErrorMsg;
        LeaveSub;
    EndIf

```

```

// Attempt to open the database.
Set ErrorCode = 0;
Run vDataSource.Open;
If vDataSource.ErrorCode not= 0
    MsgBox 'The datasource '+vDsnName+' could not be opened.
'+CHARLF+CHARLF+
        'Database error code:
'+vDataSource.ErrorCode+CHARLF+
        'Database error message:
'+vDataSource.OBCErr+ '
        Mode(Warn);
// If the database could not be opened, set an error code.
Set ErrorCode = -1;
EndIf
ENDSUB
// -----
// -----
SUB Generate_Output
    SET vWritePgf = vCurrPgf;
    NEW Paragraph NewVar(CurrPgfNew) PrevObject(vWritePgf);
    SET vWritePgf = CurrPgfNew;
    GET String FromString(v_Value) NewVar(v_Output)
        ReplaceAll(CHARLF) With(' ');
        ReplaceAll(CHARCR) With(' ');
    NEW Text Object(vWritePgf) v_Output;

//Format paragraph style according to v_Format_To_Use
    GET Object Type(PgfFmt) Name(v_Format_To_Use)
NewVar(vBodyFmt);
    SET vWritePgf.Properties = vBodyFmt.Properties;
    SET vCurrPgf = vWritePgf;
ENDSUB
// -----
// -----
SUB ParsePhoneNum
    Local vAreaCode;
    Local vExchange;
    Local vNumber;
    If vRawNum.Size = 10
        Get String FromString(vRawNum) EndPos(3)
NewVar(vAreaCode);
        Get String FromString(vRawNum) StartPos(4) EndPos(6)
NewVar(vExchange);
        Get String FromString(vRawNum) StartPos(7)
NewVar(vNumber);
        Set vFormattedNum = '(' + vAreaCode + ')' + vExchange +
        '-' + vNumber;
    Else

```

```
        Set vFormattedNum = vRawNum;
    EndIf
//
ENDSUB
// -----
```

◆◆ Automated Publishing with MML

MML is short for Maker Markup Language. MML is a very simple “tagging” structure that allows text files to be mapped to the formats of a FrameMaker template. This method of automated publishing is best used for a very simple layout. MML can only contain paragraph and character formats. If tables are necessary in the output, MML will not suffice.

As with any markup language, time and support will need to come from the IT department (or whoever maintains the database). The markup will need to be added as the data is exported from the database and written out to the MML text file. Below is a sample MML file for the directory sample used in this discussion. The MML assumes that an existing FrameMaker template exists, with the formats defined. To view the process of setting up a query or a set of queries to output MML from a database, see [Sample SQL Query that Returns MML Syntax on page 48](#).

After the query is run to add the MML markup to the data, the data can be saved as a text file from Access (or you can add further code to export it automatically). The resulting MML file can then be imported into your FrameMaker template. The MML tags “map” to the paragraph and character formats in your template. MML supports only paragraph and character formats, so it is only a valid option for database publishing the simplest projects.

FrameMaker is shipped with a complete MML reference guide in PDF format. This guide is located in the FrameMaker install directory in a subdirectory called OnlineManuals:

```
Adobe FrameMaker7.0\OnlineManuals\MML_Reference.pdf
```

FrameMaker is also shipped with a complete set of sample MML files. They are located in the FrameMaker install directory in a directory called Samples:

```
Adobe FrameMaker7.0\Samples\MML\
```

● Sample SQL Query that Returns MML Syntax

The following SQL statement demonstrates the method of “wrapping” MML markup code around data elements for importing into an existing FrameMaker template. This particular SQL statement outputs the necessary MML markup for each firm.



Keep in mind that the syntax returned by this query is correct MML syntax. However, all of the logic that puts the entries together is missing. You would either have to pre- or post-process the MML file to correctly link the executives in the directory or handle the SIC numbers and descriptions. This will further add to the complexity of using markup such as this.

```
SELECT "<FML\ Number><Emphasis>F-M-L <Default Para
Font>" + FML_Number,
"<Firm\ Name>" + Name,
"<Marker <MType 2> <MText '"+Name+"'> >",
"<TradeName>" + TradeName,
"<Firm\ Info>" + Street,
"<Firm\ Info>" + City + ", " + State + " " + ZipCode_5Digit + "-"
+ ZipCode_4Digit,
"<Firm\ Info><Emphasis>Phone: <Default Para Font> (" +
left (Phone, 3) + ") " + mid (Phone, 4, 3) + "-" + right (Phone, 4)
FROM firms;
```

Directory Entry in MML Format

The following code contains the MML markup for a single entry in the directory:

```
<FML\ Number>F-M-L 066778077
<Firm\ Name>AMERICAN ASSOCIATION OF RETIRED PERSONS
<Marker <MType 2> <MText `AMERICAN ASSOCIATION OF RETIRED
PERSONS'> >
<TradeName>Trade Name AARP
<Firm\ Info>601 E St Nw
<Firm\ Info>Washington, DC 20049-0001
<Firm\ Info>Phone (202)-434-2277
<Exec>Norman Stengel\tPr
<Exec>Richard Henry\tCOO
<Exec>Clifford J Rose\tCont
<Exec>Steven S Zaleznick\tGC
<Exec>Horace B Deets\tEx Dir
<Exec>James R Holland\tCommctns
<Exec>James A Maigret\tFin Plng
<Exec>Charles W Allen\tPubls
<Exec>Constance Swank\tRsch
<Exec>Thomas C Nelson\tSvcs
```

```
<Exec>Wayne F Haefer\tDir
<Exec>C Anne Harvey\tDir
<Exec>Mary Anne RiesenbergtDir
<Exec>Johc C Rother\tDir
<Exec>Jan Stefanov\tDir
<Firm\ Info>\rSIC 8399: health systems agency; social change
association; social service information exchange.
<Marker <MType 0> <MText `AMERICAN ASSOCIATION OF RETIRED
PERSONS'> >
```

◆◆ Automated Publishing with XML

XML stands for eXtensible Markup Language. This markup is more complex than MML and requires opening and closing tags. This section will only deal with XML as it relates to database publishing. To view the process of setting up a query or a set of queries to output XML from a database, see [Sample SQL Query that Returns XML Syntax on page 51](#).

XML is becoming the standard for data exchange. Using a markup language such as XML can be a reasonable method of database publishing. Keep in mind that the XML generated by the database will need to be valid XML. A structured FrameMaker template and application will then need to be created in order for FrameMaker to be able to import the XML and format it.

FrameMaker 7 is shipped with a complete XML cookbook in PDF format. The PDF file is located in the FrameMaker install directory in a directory called **XMLCookbook**. This cookbook and sample files demonstrate how to import and export valid XML in FrameMaker:

```
Adobe FrameMaker7.0\XMLCookbook\UserGuide.pdf
```

● Sample SQL Query that Returns XML Syntax

The following SQL statement demonstrates the method of “wrapping” XML markup code around data elements for importing into an existing FrameMaker template. This particular SQL statement outputs the necessary XML markup for each firm. The SQL is executed within Access to return data wrapped in XML. Access 2002 and other more recent database applications allow for direct XML export of data.



Keep in mind that the syntax returned by this query is correct XML syntax. However, all of the logic that puts the entries together is missing. You would either have to pre- or post-process the XML file to correctly link the executives in the directory or handle the SIC numbers and descriptions. This will further add to the complexity of using markup such as this.

```
SELECT "<FML-Number><Emphasis>F-M-L
</Emphasis>"+FML_Number+"</FML-Number>",
"<Firm-Name>"+Name+"</Firm-Name>",
"<TradeName>"+TradeName+"</TradeName>",
"<Firm-Info>"+Street+"</Firm-Info>",
"<Firm-Info>"+City+", "+State+" "+ZipCode_5Digit+"-
"+ZipCode_4Digit+"</Firm-Info>",
"<Firm-Info><Emphasis>Phone: </Emphasis> (" + left (Phone,3)+")
"+mid (Phone,4,3)+"-"+right (Phone,4)+"</Firm-Info>"
FROM firms;
```

Directory Entry in XML Format

The following code contains the final XML markup for a single entry in the directory:

```
<FML-Number><Emphasis>F-M-L </Emphasis>066778077</FML-Number>
<Firm-Name>AMERICAN ASSOCIATION OF RETIRED PERSONS</Firm-Name>
<TradeName>Trade Name <TradeName>AARP</TradeName></TradeName>
<Firm-Info>601 E St Nw</Firm-Info>
<Firm-Info>Washington, DC 20049-0001</Firm-Info>
<Firm-Info><Emphasis>Phone </Emphasis>(202)-434-2277</Firm-Info>
<Exec>Norman StengelPr</Exec>
<Exec>Richard HenryCOO</Exec>
<Exec>Clifford J RoseCont</Exec>
<Exec>Steven S ZaleznickGC</Exec>
<Exec>Horace B DeetsEx Dir</Exec>
<Exec>James R HollandCommctns</Exec>
<Exec>James A MaigretFin Plng</Exec>
<Exec>Charles W AllenPubls</Exec>
<Exec>Constance SwankRsch</Exec>
<Exec>Thomas C NelsonSvcs</Exec>
```

```
<Exec>Wayne F HaeferDir</Exec>
<Exec>C Anne HarveyDir</Exec>
<Exec>Mary Anne RiesenberDir</Exec>
<Exec>Johc C RotherDir</Exec>
<Exec>Jan StefanovDir</Exec>
<Firm-Info><Emphasis></Emphasis></Firm-Info>
<Firm-Info><Emphasis>SIC </Emphasis><SIC-NUMBER>8399</SIC-
NUMBER>: health systems agency; social change association; social
service information exchange.</Firm-Info>
```

◆◆ Automated Publishing with MIF

Maker Interchange Format (MIF) is the ASCII version of a FrameMaker file. When saving a FrameMaker document, FrameMaker usually saves the file in a proprietary binary format that only FrameMaker can understand. When saving a FrameMaker document as a MIF file, FrameMaker encodes the document in markup and saves it in ASCII format. The markup is very complex and can support any feature/function of FrameMaker. This markup can be used by other programs to create FrameMaker files.

When you do database publishing with MIF, the MIF syntax is wrapped around the data elements. A very large text (ASCII) file is then generated from the database and then is subsequently opened or imported into the final FrameMaker template. MIF files are very verbose and can be difficult to read and troubleshoot. MIF can be combined with MML to make the files easier to read and understand. To view the process of setting up a query or set of queries to output MIF from a database, see [Sample SQL Query that Returns XML Syntax on page 51](#).

By using MIF in database publishing, you can create a document almost as complex as if you had taken a data-driven approach, with a few exceptions. Because any feature in FrameMaker has an equivalent in MIF, the entire document (including the template) can be generated automatically from the database. The biggest drawback to this solution is the time required (almost all of it on the database end) to create the SQL to generate the initial MIF syntax. Referring to [Sample SQL Query that Returns MIF Syntax on page 54](#), notice the amount of syntax required just for a simple entry in the directory.

FrameMaker is shipped with a complete MIF reference guide in PDF format. The PDF file is located in the FrameMaker install directory in a directory called OnlineManuals:

```
Adobe FrameMaker7.0\OnlineManuals\MIF_Reference.pdf
```

FrameMaker is also shipped with a complete set of sample MIF files. They are located in the FrameMaker install directory in a directory called Samples:

```
Adobe FrameMaker7.0\Samples\MIF\
```

● Sample SQL Query that Returns MIF Syntax

The following SQL statement demonstrates the method of “wrapping” MIF markup code around data elements for importing into an existing FrameMaker template. This particular SQL statement outputs the necessary MIF markup for each firm.



Keep in mind that the syntax returned by this query is correct MIF syntax. However, all of the logic that puts the entries together is missing. You would either have to pre- or post-process the MIF file to correctly link the executives in the directory or handle the SIC numbers and descriptions. This will further add to the complexity of using markup such as this.

```
SELECT
"<Para <Pgftag 'FML Number'><ParaLine <Font <Ftag 'Emphasis'> >
<String 'F-M-L ' > <Font <Ftag ''> > <String '"+FML_Number+"'> >
>",
"<Para <Pgftag `Firm Name'><ParaLine <String `"+Name+"'> >",
"<Marker <Mtype 2> <Mtypename `Index'> <Mtext `"+Name+"'>
<Mcurrpage `1'> > >",
"<Para <Pgftag `TradeName'> <ParaLine <String `Trade Name'> <Char
EmSpace> <Font <Ftag `TradeName'> > <String `"+TradeName+"'> >
>",
"<Para <Pgftag `Firm Info'> <ParaLine <String `"+Street+"'> > >",
"<Para <Pgftag `Firm Info'> <ParaLine <String `"+City+",
"+State"'> <Char EnSpace> <String `"+ZipCode_5Digit+"'> <Char
EnDash> <String `"+ZipCode_4Digit+"'> > >,
"<Para <Pgftag `Firm Info'> <ParaLine <Font <Ftag `Emphasis'>
<FLocked No> > <String `Phone'> <Char EnSpace> <Font <Ftag ``'>
<FLocked No> > <String `("+ left (Phone, 3)+") "+mid(Phone, 4, 3)+"-
"+right(Phone, 4)+"'> > >
FROM firms;
```

Directory Entry in MIF Format

The following code contains the MIF markup for a single entry in the directory:

```
<Para
  <Pgftag `FML Number'>
  <ParaLine
    <Font
      <Ftag `Emphasis'>
      <FLocked No>
    > # end of Font
    <String `F-M-L ' >
    <Font
      <Ftag ``'>
```

```

    <FLocked No>
  > # end of Font
  <String `066778077'>
> # end of ParaLine
> # end of Para
<Para
  <Pgftag `Firm Name'>
  <Pgftag Referenced Yes>
  <ParaLine
    <String `AMERICAN ASSOCIATION OF RETIRED '>
  > # end of ParaLine
  <ParaLine
    <String `PERSONS'>
    <Marker
      <MType 2>
      <MTypeName `Index'>
      <MText ` AMERICAN ASSOCIATION OF RETIRED PERSONS '>
      <MCurrPage `1'>
    > # end of Marker
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `TradeName'>
  <ParaLine
    <String `Trade Name'>
    <Char EmSpace>
    <Font
      <FTag `TradeName'>
      <FLocked No>
    > # end of Font
    <String `AARP'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Firm Info'>
  <ParaLine
    <String `601 E St Nw'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Firm Info'>
  <ParaLine
    <String `Washington, DC'>
    <Char EnSpace>
    <String `20049'>
    <Char EndDash>
    <String `0001'>
  > # end of ParaLine
> # end of Para
<Para

```

```

<Pgftag `Firm Info'>
<Paraline
  <Font
    <Ftag `Emphasis'>
    <FLocked No>
  > # end of Font
  <String `Phone'>
  <Char EnSpace>
  <Font
    <Ftag ``'>
    <FLocked No>
  > # end of Font
  <String `(202)-434-2277'>
  > # end of Paraline
> # end of Para
<Para
  <Pgftag `Exec'>
  <Paraline
    <String `Norman Stengel'>
    <Char Tab>
    <String `Pr'>
  > # end of Paraline
> # end of Para
<Para
  <Pgftag `Exec'>
  <Paraline
    <String `Richard Henry'>
    <Char Tab>
    <String `COO'>
  > # end of Paraline
> # end of Para
<Para
  <Pgftag `Exec'>
  <Paraline
    <String `Clifford J Rose'>
    <Char Tab>
    <String `Cont'>
  > # end of Paraline
> # end of Para
<Para
  <Pgftag `Exec'>
  <Paraline
    <String `Steven S Zaleznick'>
    <Char Tab>
    <String `GC'>
  > # end of Paraline
> # end of Para
<Para
  <Pgftag `Exec'>
  <Paraline

```

```
<String `Horace B Deets'>
<Char Tab>
<String `Ex Dir'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `James R Holland'>
<Char Tab>
<String `Commctns'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `James A Maigret'>
<Char Tab>
<String `Fin Plng'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `Charles W Allen'>
<Char Tab>
<String `Publs'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `Constance Swank'>
<Char Tab>
<String `Rsch'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `Thomas C Nelson'>
<Char Tab>
<String `Svcs'>
> # end of ParaLine
> # end of Para
<Para
<Pgftag `Exec'>
<ParaLine
<String `Wayne F Haefer'>
<Char Tab>
```

```

    <String `Dir'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Exec'>
  <ParaLine
    <String `C Anne Harvey'>
    <Char Tab>
    <String `Dir'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Exec'>
  <ParaLine
    <String `Mary Anne Riesenber'>
    <Char Tab>
    <String `Dir'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Exec'>
  <ParaLine
    <String `Johc C Rother'>
    <Char Tab>
    <String `Dir'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Exec'>
  <ParaLine
    <String `Jan Stefanov'>
    <Char Tab>
    <String `Dir'>
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Firm Info'>
  <ParaLine
    <Font
      <Ftag `Emphasis'>
      <FLocked No>
    > # end of Font
  > # end of ParaLine
> # end of Para
<Para
  <Pgftag `Firm Info'>
  <PgfReferenced Yes>
  <ParaLine
    <Font
      <Ftag `Emphasis'>

```

```
<FLocked No>
> # end of Font
<String `SIC'>
<Char EnSpace>
<Font
  <FTag `SIC NUMBER'>
  <FLocked No>
> # end of Font
<String `8399'>
<Font
  <FTag ``'>
  <FLocked No>
> # end of Font
<String `: health systems agency; social change association;
'>
> # end of ParaLine
<ParaLine
  <String `social service information exchange.'>
  <Marker
    <MType 0>
    <MTypeName `Header/Footer $1'>
    <MText `AMERICAN ASSOCIATION OF RETIRED PERSONS'>
    <MCurrPage `1'>
  > # end of Marker
> # end of ParaLine
> # end of Para
```

■ Summary

Database publishing is an emerging marketplace. Many applications can be used to publish information from one or more databases and static files with FrameMaker. The example given in this chapter was very simple in order to better explain the different methods and applications that can be used. Obviously, much more complex publications can be generated by some of these tools.

The reason the author tends to favor a more data-driven approach like PatternStream or FrameScript is because the formatting is done at the same time. As I acquire data from the database, I can literally watch the data get formatted on the screen. When using a markup language, I must first export the data as valid XML and then set up a structured FrameMaker template and application to read the XML in and format it. The time spent creating the valid XML and structured FrameMaker template is generally more than is required in the data-driven approach. Changing the design is also much faster in the data-driven approach.

Visit the websites listed in the section [Database Publishing Applications on page 4](#) to request a demo version of these applications to see which product might best suit your specific needs. The author would also be happy to answer any questions you may have regarding getting started in database publishing. Drop Brad Anderson a message or send feedback with the form on the last page of this PDF.

■ Send Feedback

First Name

Last Name

Company Name

Address

City

State

Postal Code

Country

Email Address

URL (Website Address)

Comments/Feedback