**About the Data Access Manager** Querying other applications

Your application can use the **Data Access Manager** to gain access to data in another application and to provide templates to be used for data transactions.

The **Data Access Manager** is available with System 7.0. Use the **Gestalt** function described in **Compatibility Guidelines** to determine whether the **Data Access Manager** is present.

The **Data Access Manager** allows your application to communicate with a database or other data source even if you do not know anything about databases in general or the specific data source with which the users of your software will be communicating. All your application needs is a few high-level **Data Access Manager** functions and access to a file called a **query document.** The query document, provided by another application, contains commands and data in the format appropriate for the database or other data source. The string of commands and data sent to the data source are referred to as a **query.** Note that a query does not necessarily extract data from a data source; it might only send data or commands to a database or other application.

The **Data Access Manager** makes it easy for your application to communicate with data sources. You need only add a menu item that opens a query document, using a few standard **Data Access Manager** functions to implement the menu selection. Users of your application can then gain access to a database or other data source whenever they have the appropriate query documents. A user of a word-processing program might use this feature, for example, to obtain access to archived material, dictionaries in a variety of languages, or a database of famous quotations. A user of a spreadsheet program might use a query document to obtain tax records, actuarial tables, or other data. A user of an art or CAD program might download archived illustrations or designs. And for the user of a database application for the Macintosh computer, the **Data Access Manager** can provide the resources and power of a mainframe database.

The **Data Access Manager** also provides a low-level interface for use by applications that are capable of creating their own queries and that therefore do not have to use query documents.

If your application uses only the high-level interface and relies on query documents created by other programs, then all the routines you need to know are described in the **Data Access Manager**. However, if you want to create a query document or an application that uses the low-level interface, then you must also be familiar with the command language used by the data server.

You need the information in the **Data Access Manager** if you want your application to be able to gain access to data in other applications or if you want to write a query document.

> **Note:** The **Data Access Manager** makes it easy for your application to communicate with a database running on a remote computer. However, there is no reason why the database could not be local-that is, running on the same computer as your application. To implement such a system, you would have to have a database that runs on a Macintosh computer and that has a command-language interface, plus a database extension that can use that command

language. In most cases, it would be much simpler to run the database as a separate application and use the Clipboard to transfer data into and out of the database.

Note also that the program containing the data need not be a database. With the appropriate database extension, your application could read data from a spreadsheet, for example, or any other program that stores data.

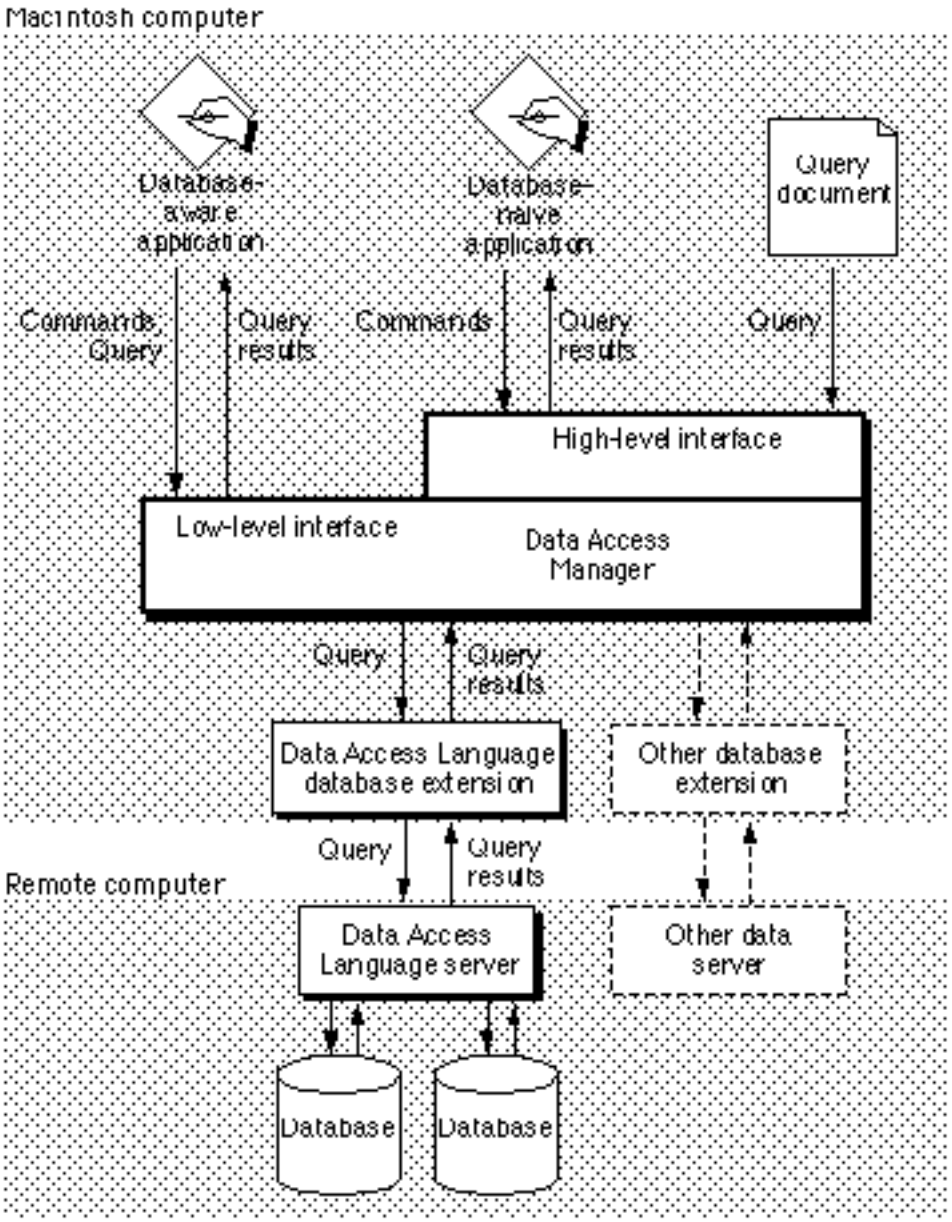
Apple provides a database extension that uses Data Access Language (DAL). A **database extension** provides an interface between the **Data Access Manager** and the database or other program that contains the data. If you want to write an application that uses the low-level interface to communicate with a Data Access Language server, or if you want to create a query document that uses Data Access Language, you must be familiar with that language. *Data Access Language Programmer's Reference,* available from APDA, fully describes this language.

The **Data Access Manager** constitutes a standard interface that allows Macintosh applications to communicate with any number of databases or other data sources through a variety of data servers. As used in this and related sections , a **data server** is the application that acts as an interface between the database extension on the Macintosh computer and the data source, which can be on the Macintosh computer or on a remote host computer. A data server can be a database server program, such as a Data Access Language server, which can provide an interface to a variety of different databases, or it can be the data source itself, such as a Macintosh application.

The **Data Access Manager** has two application interfaces: the high-level interface and the low-level interface. If the proper database extension and query documents are available in the user's system, you can use the high-level interface to communicate with a data source without having any knowledge of the command language that the data server uses. Even if you use the low-level interface, your application can isolate the user from any specific knowledge of the data source or the data server's command language.

This section presents an overview and description of the **Data Access Manager**, including diagrams and conceptual descriptions of the components and processes involved in using the high-level and low-level interfaces. Next, **Using the Data Access Manager** includes descriptions, flowcharts, and program fragments that provide a step-by-step guide to the use of the high-level and low-level interfaces.
**Creating a Query Document** describes the contents and function of a query document. You do not have to read this section unless you are writing an application that creates query documents, although if you are using the high-level interface you might be interested to know just how a query document works.

The Figure below illustrates connections between Macintosh applications and a database on a remote computer. The arrows in the Figure show the flow of information, not the paths of commands or control signals. See the two succeeding Figures for the sequences involved in sending and retrieving data.




A connection with a database

## The High-Level Interface

As the Figure above shows, a database-naive application-that is, one that cannot prepare a query for a specific data server-uses the **Data Access Manager**'s high-level routines to communicate with a data server. Because the application cannot prepare a query, it must use a query document to provide one. A query document can contain code, called a **query definition function,** that prompts the user for information and modifies the query before the **Data Access Manager** sends it to the data server. The exact format of a query definition function is described in the section entitled **Writing a Query Definition Function**.

**Note:** The term *query* refers to any string of commands (and

associated data) that can be executed by a data server. A query can send data to a data source, retrieve data from a data source, or reorganize the data in a data source. The **Data Access Manager** does not interpret or execute the query; it only implements the interface (sometimes called the *application program interface,* or API) that allows you to send the query to the data server.

When you want to use the high-level routines to execute a query on a data server, you first select a query document or allow the user to select one. You use high-level routines to

- get the query from the query document
- execute the query definition function to modify the query
- send the query to the data server
- retrieve the results from any query that asks for information from the data source
- convert to text the results returned by a query

For example, suppose a company that makes rubber ducks has a database on a minicomputer that contains a mailing list of all its customers. The database has a Data Access Language interface and the company's marketing manager has a Macintosh computer with an application that uses high-level **Data Access Manager** routines to communicate with the remote database server. As the Figure below illustrates, the marketing manager must also have a query document, created by another application, that she can use to get an address from the mailing list on the remote minicomputer. The query document can be as complex or as simple as its creator cares to make it; in this example, the query document is designed specifically to obtain addresses from the rubber duck mailing list. The marketing manager might have several other query documents available as well: one to extract a mailing list for a specific zip code, one to list all of the customers who have made a purchase within the last year, and so on.

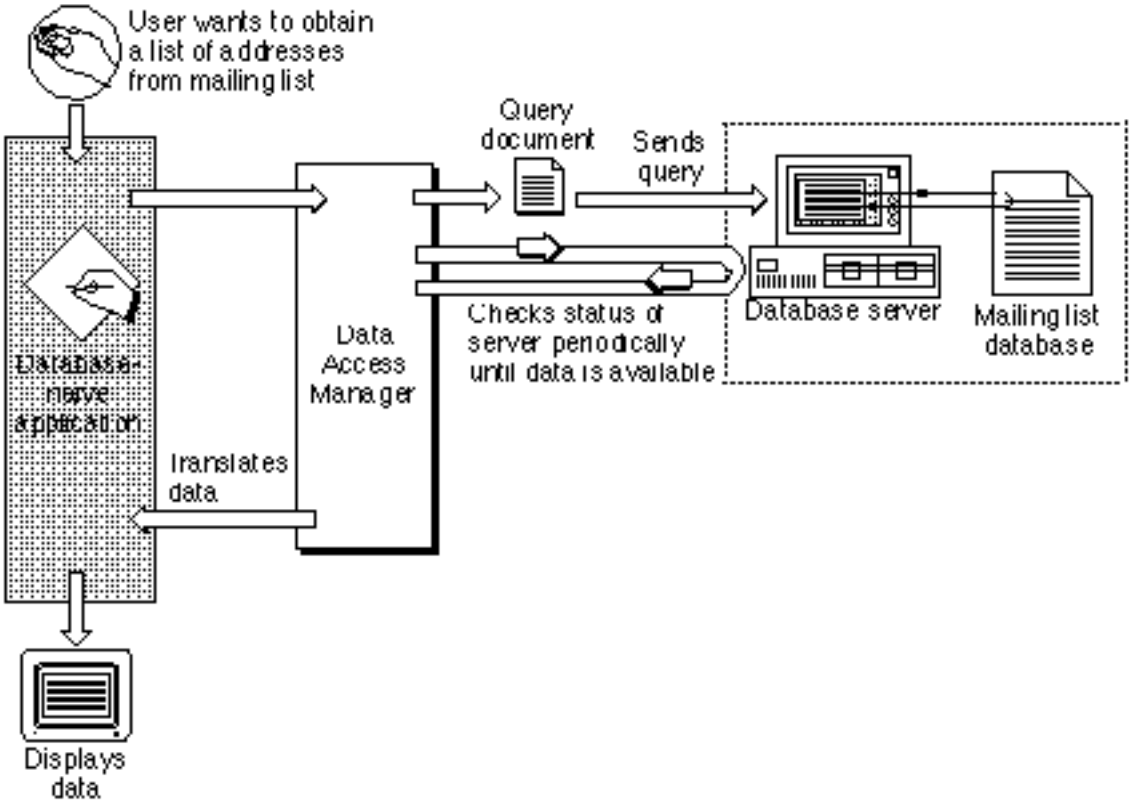
Notice that once the query document has sent the query to the data server, the **Data Access Manager** handles the data retrieval. Although query documents and high-level **Data Access Manager** routines make it very easy for you to *request* data from a data source, there is no way for a query document to verify that data *sent* to a data source has been successfully received. For that reason, it is recommended that you use the low-level interface to send data to a data source or update data in a data source.

## Sending a Query Through the High-Level Interface

To obtain a list of addresses from the mailing list, the marketing manager chooses the **Open Query** menu command from the **File** menu in her application. From the list of query documents displayed, she chooses one named Rubber Duck Address List.

The application calls the **Data Access Manager** function **DBGetNewQuery**. This function opens the Rubber Duck Address List query document and creates a

partial query from the information in the query document. The partial query specifies the type of data (character strings) and the columns from which the data items should come (the name and address columns). The partial query lacks some specific data (the rows that should be searched) that is needed to complete the search criteria.




Using high-level **Data Access Manager** routines

Next, the application calls the **DBStartQuery** function, which in turn calls the query definition function in the query document. The query definition function displays a dialog box that asks for the purchase dates to search. When the marketing manager types in the requested information and clicks OK, the query definition function adds the data to the partial query in memory. The query is now ready to be executed.

Next, the **DBStartQuery** function sends the query to the Data Access Language database extension, and the database extension sends the query over a communications network to the remote Data Access Language server. Finally, the **DBStartQuery** function commands the Data Access Language server to execute the query.

## Retrieving Data Through the High-Level Interface

When the application is ready to retrieve the data that it requested from the database, the application calls the **DBGetQueryResults** function. This function determines when the data is available, retrieves it from the data server, and places the data in a record in memory. The application can then

call the **DBResultsToText** function, which uses routines called **result handlers** to convert each data item to a character string. The **DBResultsToText** function passes to the application a handle to the converted data. The application then displays the list of customers for the marketing manager.

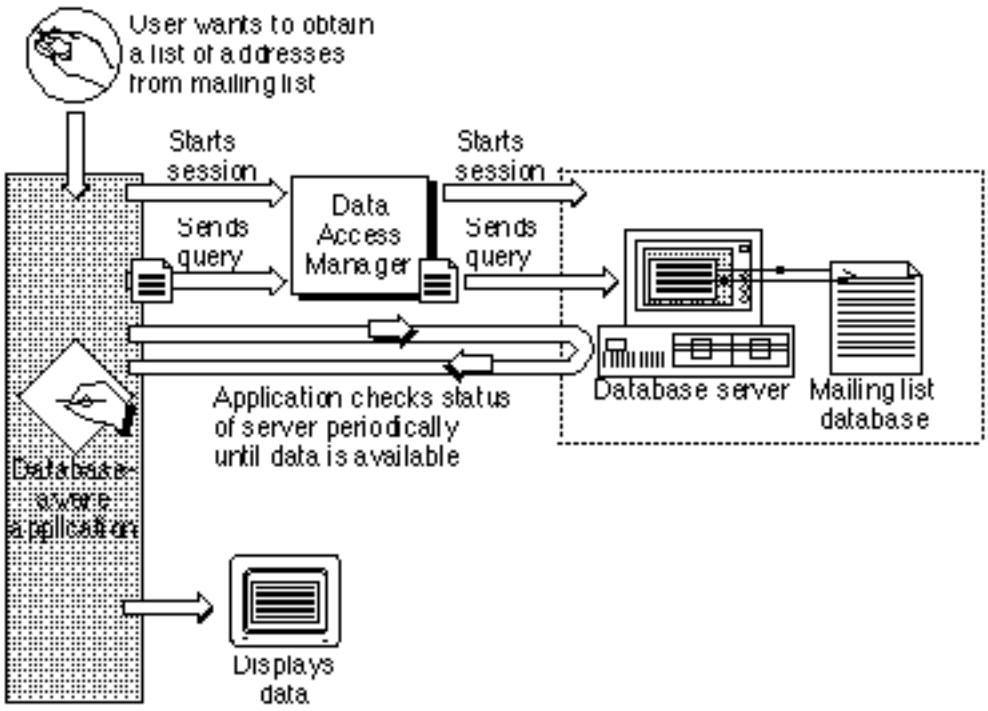
Data items and result handlers are described in the section entitled **Processing Query Results**

## The Low-Level Interface

A database-aware application communicates through the low-level interface of the **Data Access Manager**. You can use the low-level interface to

- initiate communication with the data server, sending the user name, password, and other information to the data server
- send a query to the data server
- execute the query that you have sent to the data server
- halt execution of the query
- return status and errors from the data server
- send data to the data source
- retrieve data from the data source

For example, suppose once again that a company that makes rubber ducks has a mailing list of all of its customers in a database on a minicomputer, and the database has a Data Access Language interface. This time, suppose the Macintosh application the marketing manager is using calls low-level **Data Access Manager** routines to communicate with the remote database server. The Figure below illustrates the use of the low-level interface. Notice that if you use the high-level interface (as shown in the preceeding Figure ), the query document and the **Data Access Manager** prepare the query, send the query, retrieve the query results, and translate the data for you. If you use the low-level interface, however, you must perform these functions yourself.




Using low-level **Data Access Manager** routines

## Sending a Query Through the Low-Level Interface

To update the mailing list with a new address for customer Marvin M., the marketing manager enters the new address into her application. The application prepares a Data Access Language statement (a query) that specifies the type of data (a character string), the column into which the data item should go (the address column), the row to be modified (the Marvin M. row), plus the actual data the application wishes to send (Marvin M.'s address). The application then passes this query to the **Data Access Manager** using the low-level interface. (The application can send the query in several pieces or all at once.) The **Data Access Manager** sends the query to the Data Access Language database extension in the Macintosh computer, and the database extension sends the query to the remote Data Access Language server.

## Retrieving Data Through the Low-Level Interface

Once the query begins executing, the application can periodically check with the data server to determine whether the data is ready (as shown in the Figure above). When the data is available, the application must retrieve it one data item at a time. An application that uses the low-level interface must determine the data type of each data item, convert the data into a format that is meaningful to the user, and store the data in memory allocated by the application. Data types are described under **Getting Query Results** in the section entitled **Processing Query Results** .

Note that neither the **Data Access Manager** nor the DAL database extension reads, modifies, or acts on the query that an application sends to the data server. The data server does execute the query, causing the data source to accept new data or prepare data for the application. To use the low-level interface to communicate with a data server, your application must be capable of preparing a query that can be executed by the data server.