Choosing a Client-Server Report Writer

Martin Knapp
JetForm (Proactive Division)
Four, Main Street
Los Altos
CA 94022
Tel: (415)-949-9100

Introduction

What is client-server?

It may seem a bit odd to begin a talk on client-server by asking such apparently basic questions. However, one of the problems with discussing client-server is that theoretical debate can sometimes get in the way of practical applications. This is not to say that theoretical debate is unnecessary. But in our case, we are interested in a relatively limited subset of client-server, so we will concentrate here on a practical working implementation.

User Expectations

It is essential to understand what users expect of a client-server reporter before choosing one to implement, since the ability to live up to user expectations is one of the definitions of a quality solution. And these expectations have been moulded over the last decade by the evolution of the PC, and Windows.

What is a Report Writer?

As we will see, report writers do not all fall into the same category. So we need to consider exactly what it is we are seeking to achieve, in order to make a choice that best satisfies our requirements.

Management Issues

In choosing our software, we also need to consider issues involved in managing what is in effect a new kind of application, which allows users an unprecedented freedom of access to company data.

ODBC and Middleware

One thing that all client-server applications have in common is the connectivity issue: how do we link the client to the server? We will look briefly at the different possibilities in this field, including some problems specific to the HP3000.

What is client-server?

Theoretical definitions

This is not the place for a detailed exposition of client-server theory. However, there are a few points which are worth bearing in mind here, since they avoid us taking a too narrow view of the question. Above all, they allow us to understand the implementation of a client-server report writer as merely the first step down a much longer road, which will certainly lie at the heart of computing in the future.

The first point to remember, then, is that “client-server” does not necessarily have any hardware or telecommunications implications. It is quite possible to implement client-server on a single system, where one process acts as a “client”, issuing requests for another process to “service”.

The second point is that client-server is not a one-to-one relationship. A single server may - usually does - have many clients. By the same token, a client may make requests of many servers.

Thirdly, in complex applications where client requests are satisfied by many servers, it may be necessary to adopt a multi-tier strategy. Rather than the client making requests directly of one or several servers, it will make its requests of a broker, which can then despatch them to the appropriate servers.

Practical definitions

A theoretical definition is vital to show us where we are going. In reality, we rarely have the opportunity to install systems on a greenfield site. We have to integrate the new with what already exists, both in terms of computer hardware, software, and telecommunications, and perhaps more importantly in terms of what users expect and are trained to use. This means

that for practical purposes, a first implementation of client-server means **integrating existing PCs** with corporate databases held on servers.

A bit of History

Fig 1

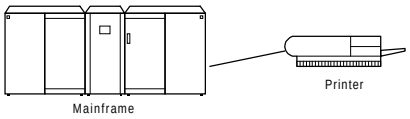
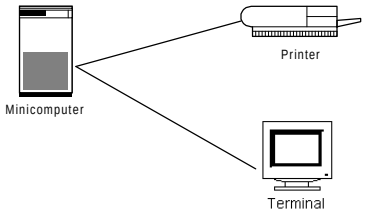


Fig 2



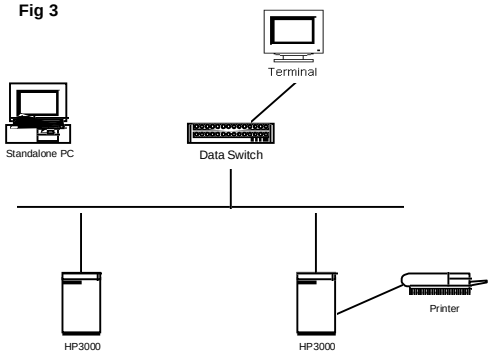
It is always easier to understand where we are, and where we are going, if we first understand where we have come from. So our readers will forgive us if we indulge in a little history!

Fig 1 shows us the early stages of commercial computing. All the input is by punched card, and the user only ever accesses data in paper form. The computer is locked away in a "holy of holies" attended by white-coated acolytes.

In Fig 2, we can see a typical mini-computer installation, in the days before LANs became common. Now the user can access data via a terminal, directly from his own desktop. If the printer is on the desktop as well, or near it, then the user can also get printed output direct, without being dependent on the operations department.

We could in fact consider this to be the beginning of the process that leads to today's client-server applications. Data no longer "belongs" exclusively to the operations department. It is being brought back onto the user's desktop where it belongs, while operations takes care of ...operating!

Fig 3



In Fig 3, we have a fairly typical proprietary minicomputer network (in this case, they are HP3000s). From the user's viewpoint they change is not spectacular. He still has his desktop terminal and printer. In this network, however, there is one big difference: using a data switch, he can connect to either host system, and run applications on each. We also notice that a standalone PC has entered the department. It's probably running a word processor and a spreadsheet.

Now all this is a definite step forward, but from the user's point of view it is still cumbersome. He cannot get data from both HP3000's on the screen at the same time, still less combine the data. If he wants to do "what-if?" operations on his data using the spreadsheet, then he has to jot down figures displayed on the screen, and re-enter it on his PC.

Time passes. The company changes. New technology emerges, at lower and lower prices. Applications are no longer written in-house, the company prefers to buy in application packages, and if necessary the hardware to run them on as well. The amount of company data not held on computer is trivial. Now, when a user talks about "consulting a file", he is more likely than not consulting a computer-based file. Books of listings begin to be replaced by on-screen interrogation, and print on demand.

We've attached the standalone PC's to the network (and they are no longer XT's, but 486s or Pentiums running sophisticated Windows applications). The printer is also on the network, so that it can be addressed both from the PCs and the servers. A Netware PC server has been added, to provide some extra disk space, and several Unix boxes (in our figure, we can see an HP-UX and an IBM AIX system, but there are probably some other ones which we've not had room to draw!) have also made an appearance. However, even the PC user still has to access his data via a terminal emulator, and integrating data from different servers with each other, or with his increasingly complex spreadsheet or PC database applications, is just as primitive as ever (except that now at least he can have all these applications in separate windows at the same time).

Practical client-server

As we have just seen, many of the requirements for client-server are already falling into place, so we are not approaching the problem from scratch. Most companies now have PCs connected to their servers, and use terminal emulators to access server applications. The PCs also have the typical PC applications installed: word processor, spreadsheet, in some cases a PC database such as MS-Access. The big drawback with this situation is the lack of any integration between the two.

A very significant feature of this situation is that control over what happens on the desktop is moving away from the IT department. How much this happens will vary from company to company: in some cases, for example, IT will keep a strict control over what PC software users can install. In general, however, we can say that users expect some freedom over what they do on their own PCs.

This can have big disadvantages. One, which came to light relatively early, is the tendency for user departments to substitute for IT, building large applications on the PC which are not properly backed up (for example), and which leave users spending more time on IT development than they do on their own jobs!

These problems are compounded if users are allowed to build applications using corporate data on servers where access has hitherto been strictly controlled. We will look at these in more detail later on. For the moment, it is important to be aware that IT has the right, and the responsibility, to control the way in which integration is achieved between existing PC applications and software packages, and corporate data.

Great expectations

User Expectations

When we start looking at integrating the PC environment with corporate data, we need to bear in mind that user expectations are determined by the software they are already using: word processors and spreadsheets are typical examples.

What are the expectations that such packages have given users?

Seamless access to company data

When you load a specific spreadsheet or WP document, you expect it to have a meaningful name, and to provide the data that you asked for. Corporate databases should behave in the same way. Users do not expect to see a jumble of meaningless table and column names, with no indication of how they are related.

Flexibility

PC packages allow data to be presented pretty much as the user likes. Obviously, corporate data is subject to certain limitations - we can't display data that isn't there for example! - but within those limitations the user should be able to manipulate the data as he sees fit, both in terms of appearance (fonts, graphics, etc) and presentation (sorts, totals, etc)

Ease of use

Windows packages have come a long way since the first DOS spreadsheets. Users expect all the "point and click" ease of use that they get today from the Windows interface.

What is a Report Writer?

On the face of it, the question may seem elementary. We all know what a report writer is... or do we? In fact, if we consider the vast range of products that fall more or less into the report writer category, we begin to see that they can be divided into a number of different sub-categories:

Simple data extraction

These are basically reporters designed to run on PCs, against simple PC databases. Even where they allow access to corporate data (via ODBC for example), they assume that the user understands the structure of the underlying database, and how tables can be joined to produce results other than simple lists. The majority of products fall into this category. HPInfoAccess, for example.

Executive Information Systems

These are not really report writers at all, although they may include some of the latter's features. Their main purpose is to provide highly condensed information, often in graphical format, and to allow the user to investigate the data in many different ways. Such products often demand a considerable amount of programming before they can be used, either in defining features such as "drill-down" searches, or in defining the data warehouses (usually in a proprietary format) that they need to work. Some examples: Powerplay, Forest & Trees.

Management Information Systems

These are aimed more at providing easily built and manipulated lists of data for functions ranging from stock control to sales analysis. We distinguish them from the simple list products described above, in that they are designed to work against corporate data, and as such protect the user from the complexity of the underlying data structures. Some examples: Impromptu, Fantasia Reporter

Industrial Reporting

The term “Industrial Reporting” implies printed output that is either very large, or produced on a regular basis (often both), and where the printing is carried out in batch on the server, not on the PC. In the past, such reports either had to be set up by the DP department, or were programmed using a green-screen based reporter, which was cumbersome at best. A PC-based reporter which also includes a server-based module should be able to set up reports to run completely independently of the PC and the user.
Why would we want to do this? There are a number of reasons:
Development of industrial output: for example, customer letters such as those produced by the insurance or mail-order industries for example.
Large reports: once a report reaches a certain size (about 100 pages, for example), printing it on the typical networked laser printer becomes impractical. It takes too long to print, clogs up the printer for other users, and imposes a heavy network load. It is more convenient to print on a high-capacity printer attached to the server.

Reports requiring heavy database access: a three-page report may need to query millions of rows in the target database. It clearly makes sense to run it out of working hours, in batch.

Regular reports: many reports are run on a regular basis (weekly, monthly for example). Once a user has set up such a report, he/she can do without the inconvenience of requesting the report by hand every week. With batch processing, the report can be ready and waiting when it is required.

Management Issues

Before client-server, users’ activity was safely contained. They either used green-screen programs, where data access logic was always kept within the application, and therefore strictly controlled, or PC applications where any damage was limited to a single PC. Now with a client-server reporting tool, the PC user from hell can cripple your server, by launching data requests joining multiple million-row tables - with sequential reads to boot! Or he may saturate the network by trying unwittingly to download millions of bytes of data to his PC.

Actually this is unfair on the user. The whole point of a PC reporting tool should be to allow the user to access the data he needs for his job, without worrying about its underlying technical implementation. And if we look a little further, we can see a whole series of management issues that must be considered for an installation to be successful.

Performance, we have just discussed.

Security: you may want to restrict access to data, both by table, and by content (for example, a salesman should only be able to see the sales data for his particular region).

Data coherence: without a good knowledge of the database structure, a user could build reports which ran correctly, but produced incoherent data. If anything, this is the most dangerous problem of all, since these reports will then form the basis for incorrect business decisions.

Data distribution: in many cases, databases will be distributed around a network of servers. The user should not need to know about the physical position of databases.

Conclusions

We've now finished our review of some of the issues involved in choosing and installing a client-server report writer. We can't make the final choice for you! But let's just run over again some of the criteria we have highlighted:

Decide what kind of reporting tool you need. Do you want an Executive Information System, of is Management Information closer to your requirements? In many cases, your requirements will include elements from both, so you will have to see which product provides the best mix of features at the best price.

• Look for administration functions, which will allow you:

a) to protect your server from the users

b) to protect your users from the complexity of the underlying database

In our opinion, this is a *sine qua non* of client-server reporting, and should serve to eliminate the majority of products in the market that do not provide any such functionality

Determine whether you are likely, either now or in the future, to need a batch reporting functionality. Only a very few products will satisfy this requirement.

Client-Server Connectivity

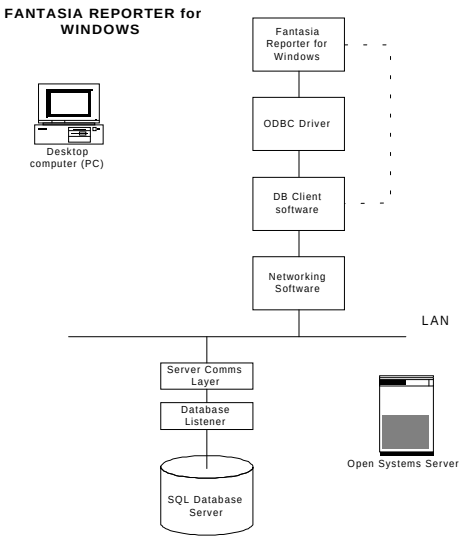
Choosing a report writer is one thing, but once it has been selected we still have to establish the connection between the client software and the database. We do not intend here to go into a detailed discussion of network hardware - since we can generally assume that this is already in place - but there are a number of issues to be resolved, which generally come under the heading of "middleware".

We will start by looking briefly at the **components** of a client-server connection, so that we know what we're talking about!

Then we'll go on to consider some issues in installing middleware, especially the part that **ODBC** has to play in all this.

Finally, we'll consider some problems that may arise if you have **multiple database types** in your network, and look at how 3rd-part middleware may help to resolve them.

Client-Server Structure



This figure gives us an overview of the different components in a simple client-server system. Setting it up is rather like playing those labyrinth games that you find in children's magazines: Pooh Bear User is on the outside of the maze, and the pot of honey in the middle. Help Pooh Bear get to the pot of honey!

First is the application. In our example it is the FANTASIA REPORTER software, but it could also be a spreadsheet like MS-Excel, or a bespoke application written in Visual Basic, or an off-the-shelf application designed to run against a particular database.

A database application on a PC is not fundamentally different from one on the host: it has to access the database by submitting SQL statements to procedures supplied with the database, commonly known as Application Program Interfaces (APIs). Naturally, every database has its own API (the DB Client software in Fig 5), and they are all different. This means that while the same SQL statement can be submitted to either Oracle or Sybase (for example) and obtain the same result, they have to be submitted through APIs which are completely unrelated. If you are developing a Visual Basic program using the Oracle APIs, for example, you will not be able to run the same program against a Sybase database without rewriting all the database calls.

This situation is obviously a big hole in the Open Systems principle, which requires that different elements of a complete system should plug together interchangeably. One solution is to use ODBC (Open DataBase Connectivity), which effectively provides a translation service between an API with a standard format on the application side, and the APIs belonging to the target database.

The next layer on our way to the honey pot, is the PC networking software. Of course, this has to be considered in conjunction with the communications software that will run on the host, and the kind of hardware connectivity you have installed. In our example, we are running over a LAN, and the chances are that we will be using the TCP/IP protocol.

Below the communications layer on the host comes the database listener, which listens out for connection requests from clients, and establishes the link between a client application and the RDBMS. Each RDBMS, of course, has its own listener software.
At last, we have reached our RDBMS honey pot!

ODBC versus proprietary
There are basically two kinds of ODBC driver:

- single tier “translators”. These sit between a client application and the native database API, and “translate” ODBC calls into appropriate database calls. This means that they require the presence of the native database software, for example (eg Oracle SQL*Net);
- multi-tier middleware. This is software that completely replaces both the native database APIs, and the native listener on the server. It is known as “multi-tier” because it covers both client and server sides. We will discuss the advantages of this later.
-

ODBC performance has had a bad press in the past, largely because:

- the ODBC standard was still in its early stages
- ODBC drivers were purely “translators”, which often had difficulty fitting ODBC calls to the “native” equivalents, and suffered as a result
- buffering requirements could cause memory problems on older PCs.

However, these failings have now been largely overcome. This is especially true for 3rd-party middleware written directly to the ODBC standard, which is able to achieve significant

performance improvements thanks to its complete control over the whole client-server process.

Depending on the **applications** you are using, there may in fact be no choice but to use ODBC. If you intend to program everything in Visual Basic, then you can use the native database APIs. However, if you want to use PC packages (eg Excel, etc), then in many cases ODBC is the only connectivity support available. In fact, ODBC has effectively established itself (for the moment anyway) as the standard access method for PC client-server applications.

Connectivity Issues

It is not uncommon today (especially in the Unix environment) for **databases from several suppliers** to be present on the same network. This raises important issues when setting up client-server connectivity. Let's take a hypothetical example of a company that uses Oracle, Sybase and Informix databases, all of which we want to access from our report writer. What problems will we encounter?

- Cost. We need to buy three different database middleware products, and this can be very expensive.
- Complexity. As well as installing separate server software for each database, we will need to install three different sets of native APIs on each PC as well. If we have chosen to use ODBC, then this will also include configuring the ODBC driver for each database type.

We can go a good way to avoiding these problems if we use 3rd-party middleware. In this case, we will need separate server modules for each database (however, the configuration will be similar in each case). We make real gains on the PC, where we need only install and configure a single client API. The advantage will be even greater if the middleware is written directly to the ODBC standard.

HP3000 issues

Allbase or IMAGE/SQL users do not have such a wide choice in middleware. However, there is one issue which is significant in some sites: the availability of a LAN. There are still many HP3000 networks which only have serial connectivity between PCs (attached to DTCs) and the server. In this case, you have no choice but to use 3rd-party middleware, since the software supplied by HP only supports LAN connectivity.

This concludes the paper. I hope that I have given you some pointers on the road to successful client-server reporting!

Martin Knapp, 17 May, 1995