Chapter – 1 The Development Environment

On any screen, you can find the transaction code of the current transaction. To do this, choose the menu path System->Status.

Display all function key (or *f-key*) settings by right-clicking anywhere in the screen area. This works on all screens. Function keys 1 through 12 are the top row of keys on your keyboard. Function keys 13 through 24 are those same keys with the Shift key held down. For example, F13 is Shift+F1. For f-keys 25 through 36, use Ctrl instead of Shift. For 37 though 48, use Alt. For f-keys higher than 48, hold down Ctrl and type the key number on the numeric keypad. For example, for F50, hold down Ctrl and type 50 on the numeric keypad.

You can learn about the functional areas by reading the R/3 online reference (R/3 menu path Help->R/3 Library). It contains tutorials and information for all of the functional areas. If you have access to a system with IDES data (International Demo and Education System), you can work through the R/3 Library exercises as well. For availability of training courses, contact SAP (http://www.sap.com) or Lambton College (http://www.lambton.on.ca).

The predecessor to R/3 is R/2. R/2 is mainframe-based, and SAP ported it to the client/server environment. To do this, SAP created Basis. Creating Basis enabled the existing ABAP/4 code to run on other platforms.

Basis is like an operating system for R/3. It sits between the ABAP/4 code and the computer's operating system. SAP likes to call it *middleware* because it sits in the middle, between ABAP/4 and the operating system.

Basis is a collection of R/3 system programs that present you with an interface. Using this interface the user can start ABAP/4 programs

To access the Basis administration tools from the main menu, choose the path Tools->Administration.

To see a list of the servers currently running in your R/3 system, choose the menu path Tools->Administration, Monitoring->System Monitoring->Servers. To view the current system log, choose Tools->Administration, Monitoring->System Log. To see system performance statistics, run transaction st03 (enter **/nst03** in the Command Field), choose This Application Server, Last Minute Load, and analyze the last 15 minutes. Press the Dialog button at the bottom of the screen, and you will see the average user response time for the last 15 minutes (look at Av. Response Time).

An *application server* is a set of executables that collectively interpret the ABAP/4 programs and manage the input and output for them. When an application server is started, these executables all start at the same time. When an application server is stopped, they all shut down together. The number of processes that start up when you bring up the application server is defined in a single configuration file called the *application server profile*.

In a three-tier client/server configuration, the presentation servers, applications servers, and database server all run on separate machines. This is the most common configuration for large systems, and is common in production.

In the two-tier client/server configuration, the presentation and application servers are combined and the database server is separate. This configuration is used in conjunction with other application servers. It is used for a batch server when the batch is segregated from the online servers. A SAPGUI is installed on it to provide local control.

Defining an R/3 System

The simplest definition of an R/3 system is "*one database.*" In one R/3 system, there is only one database. To expand the definition, R/3 is considered to be all of the components attached to that one database. One R/3 system is composed of one database server accessing a single database, one or more application servers, and one or more presentation servers. By definition, it is all of the components attached to one database. If you have one database, you have one system. If you have one system, you have one database. During an implementation, there is usually one system (or one database) assigned to development, one or more systems designated for testing, and one assigned to production.

In the most general terms, an *instance* is a *server*. It is a set of R/3 processes providing services to the R/3 system. Most of the times an Instance is referred to Application Server.

Application Server Architecture

All requests that come in from presentation servers are directed first to the dispatcher. The dispatcher writes them first to the dispatcher queue. The dispatcher pulls the requests from the queue on a first-in, first-out basis. Each request is then allocated to the first available work process. A work process handles one request at a time.

To perform any processing for a user's request, a work process needs to address two special memory areas: the user context and the program roll area. The user context is a memory area that contains information about the user, and the roll area is a memory area that contains information about the programs execution

A *user context* is memory that is allocated to contain the characteristics of a user that is logged on the R/3 system. It holds information needed by R/3 about the user, such as:

The user's current settings

The user's authorizations

The names of the programs the user is currently running

A *roll area* is memory that is allocated by a work process for an instance of a program. It holds information needed by R/3 about the program's execution, such as:

The values of the variables

The dynamic memory allocations

The current program pointer

Each time a user starts a program, a roll area is created for that instance of the program. If two users run the same program at the same time, two roll areas will exist-one for each user. The roll area is freed when the program ends.

> When speaking to a Basis consultant, you might hear the term roll area used to refer to all roll areas for one user or even all roll areas on one application server. You usually can determine the intended meaning from the context in which it is used.

Changing from the Initial Screen to the Address Screen can be considered a DIALOG STEP.

> A dialog step is used by Basis consultants as the unit of measure for system response time.

A dialog step is the processing needed to get from one screen to the next. It includes all processing that occurs after the user issues a request, up to and including the processing needed to display the next screen. For example, when the user clicks the Enter key on the Change Vendor: Initial Screen, he initiates a dialog step and the hourglass appears, preventing further input. The `sapmf02k` program retrieves the vendor information and displays it on the Change Vendor: Address screen, and the hourglass disappears. This marks the end of the dialog step and the user is now able to make another request.

There are four ways the user can initiate a dialog step. From the SAPGUI:

Press Enter.

Press a function key.

Click on a button on the screen.

Choose a menu item.

It is important for an ABAP/4 programmer to know about dialog steps because they form a discrete unit of processing for an ABAP/4 program.

An ABAP/4 program only occupies a work process for one dialog step. At the beginning of the dialog step, the roll area and user context are *rolled in* to the work process. At the end of the dialog step, they are *rolled out*.

During the roll-in, pointers to the roll area and user context are populated in the work process. This enables the work process to access the data in those areas and so perform processing for that user and that program. Processing continues until the program sends a screen to the user. At that time, both areas are rolled out. Roll-out invalidates the pointers and disassociates these areas from the work process. That work process is now free to perform processing for other requests. The program is now only occupying memory, and not consuming any CPU. The user is looking at the screen that was sent, and will soon send another request.

When the next request is sent from the user to continue processing, the dispatcher allocates that request to the first available work process. It can be the same or a different work process. The user context and roll area for that program are again rolled in to the work process, and processing resumes from the point at which it was left off. Processing continues until the next screen is shown, or until the program terminates. If another screen is sent, the areas are again rolled out. When the program terminates, the roll area is freed. The user context remains allocated until the user logs off.

In a system with many users running many programs, only a few of those programs will be active in work processes at any one time. When they are not occupying a work process, they are rolled out to extended memory and only occupy RAM. This conserves CPU and enables the R/3 system to achieve high transaction throughput.

> ABAP/4 programs do not have the capability to intercept many common Windows events. The events that generate a lot of messages such as key presses focus changes, and mouse movements are not passed to ABAP/4 programs. As a result, there is no way of performing some of the functions that are found in other Windows programs. For example, in ABAP/4, you cannot validate the contents of a field when the user presses the Tab key. You must instead wait until the user initiates a dialog step.

Understanding the Components of a Work Process

Each work process has four components

- A task handler
- An ABAP/4 interpreter
- A screen interpreter
- A database interface

All requests pass through the task handler, which then funnels the request to the appropriate part of the work process.

The interpreters interpret the ABAP/4 code. Notice that there are two interpreters: the ABAP/4 interpreter and the screen interpreter. There are actually two dialects of ABAP/4. One is the full-blown ABAP/4 data processing language and the other is a very specialized screen processing language. Each is processed by its own interpreter.

The database interface handles the job of communicating with the database.

Understanding the Types of Work Processes

There are seven types of work processes. Each handles a specific type of request. The type of work processes and the types of requests that they handle are shown in Table 1.2.

Table 1.2 Types of Work Processes and the Types of Requests they Handle

WP Type	Request Type
D (Dialog)	Dialog requests
V (Update)	Requests to update data in the database
B (Background)	Background jobs
S (Spool)	Print spool requests
E (Enqueue)	Logical lock requests
M (Message)	Routes messages between application servers within an R/3 system
G (Gateway)	Funnels messages into and out of the R/3 system

Understanding the Logon Client

The term *logon client* has nothing to do with Client/Server-it is completely different. The *logon client* refers to the number that the user types in the Client field on the logon screen

The number entered in Logon Client field (800) by the user, corresponds to a set of rows within each **client-dependent table within the database.**

Understanding Client-Dependent and Client-Independent Tables

There are two types of tables in an R/3 database: *client-dependent* and *client-independent*. A table is client-dependent if the first field is of type `CLNT`. The length will always be 3, and by convention, this field is always named `mandt`. If the first field is not of type `CLNT`, the table is client-independent.

The logon client mechanism divides the rows within a client-dependant table into distinct groups. To access a different set of data, the user logs on and specifies a different client number.

> The user master records (containing R/3 user IDs) are client-dependent. Therefore, to gain access to a client, the security administrator must create a new user ID for you within that client.

The Open SQL database statements `insert`, `update`, `modify`, and `delete` also provide automatic client handling.

> The average R/3 installation has three systems: development, test, and production. By default, each system comes with three clients installed: 000, 001, and 066. It is common to have from three to six clients in the development and test systems, but rarely will you see more than one client in production.

Using SAP's Open SQL

ABAP/4 code is portable between databases. To access the database in an ABAP/4 program you will code SAP's *Open SQL*. Open SQL is a subset and variation of ANSI SQL. The ABAP/4 interpreter passes all Open SQL statements to the database interface part of the work process (see Figure 1.27). There, they are converted to SQL that is native to the installed RDMS. For example, if you were running an Oracle database, your ABAP/4 Open SQL would be converted by the database interface to Oracle SQL statements.

If you use Open SQL, your SQL statements will be passed to the database interface. Using Open SQL has three main advantages. All of these advantages are implemented via the database interface.

Portability

The first advantage is the fact that your SQL statements will be portable between databases. For example, if for some reason your company wanted to switch from an Oracle to an Informix database, it could change the database, and your ABAP/4 code would continue to run without modification.

Buffering Data on the Application Server

Secondly, the database interface buffers information from the database on the application server. When data is read from the database, it can be stored in buffers on the application server. If a request is then made to access the same records, they would already be on the application server, and the request is satisfied from the buffer without having to go to the database. This buffering technique reduces the load on the database server and on the network link between the database and application servers, and can speed up database access times by a factor of 10 to 100 times.

Automatic Client Handling

The third advantage of using Open SQL is *automatic client handling*. With Open SQL, the client field is automatically populated by the database interface. This gives your development and testing teams many advantages, such as the ability to perform multiple simultaneous testing and training on a single database without interference from each other.

Summary

R/3 supports multiple hardware platforms, operating systems, and databases.

In addition to allowing native SQL, ABAP/4 provides Open SQL. Using Open SQL makes your code portable, faster, and provides automatic client handling.

The logon client enables multiple independent groups of data to be stored in the same table. The data you access depends on the client number you entered when you logged on.

Q **Can I copy an existing client to a new client?**

A The Basis consultant can do this for you using a client copy utility. Each development system normally has at least a reference client and a working client. Your "goodÓ data is kept in the reference client. It is copied to create the working client. If you corrupt the working client, the Basis consultant can restore it to its original state by copying the reference client again.

Q **Can I write a program that reads data from a client other than the client I am currently logged on to?**

A Yes. You can add the keywords *client specified* to any Open SQL statement. For example, to read data in client 900, you could code:

```
select * from tbl client specified where mandt = '900'.
```

However, you should realize that this should only be done if you are writing a system program. It is never done in an application program; they should always be client-independent. If you need to transmit data between two production clients, you should implement the data transfer via ALE.

/nex - Logg off. **To log off from the system without a confirmation prompt**
Enter: **/nex**. Caution: Changes that were not saved are lost without warning.
To call a transaction

in the same session (window) Enter: **/nxxxx** (xxxx = transaction code).

in the same session (window), whereby the initial screen is skipped.Enter: **/*xxxx** (x-tco)

in an additional session,Enter: **/oxxxx** (xxxx = transaction code).

To end the current transaction - Enter: **/n**.
Caution: Unsaved changes are lost without warning

To delete the current session. - Enter: **/i**.
To generate a session list - Enter: **/o**.

To end the current transaction and return to the starting menu - Enter: **/ns000**.

To log off from the system Enter: **/nend**.

Quiz

1. Choose the menu path Tools->Administration, Monitoring->System monitoring->User overview. What is the transaction code for this transaction?
2. What is the transaction code for the R/3 main menu? (The main menu is the first menu displayed after logon.)
3. What is the transaction code for the menu path Tools->Development Workbench?
4. If there are three R/3 systems in your current system landscape, how many databases are there?
5. If an R/3 system has two application servers, how many instances does it have?
6. What is Open SQL?
7. What advantages does Open SQL offer over native SQL?
8. Which part of the work process is used to implement Open SQL?
9. When is a roll area allocated, when is it de-allocated, and what does it contain?
10. When is a user context allocated, when is it de-allocated, and what does it contain?
11. When does the roll-out occur, and why does it occur?