

Database on demand

BY JIM PORELL

In a world of application servers that are shared across multiple customer constituencies, you want to make sure that data for each customer is protected from the others. Customers can get comfortable running the same applications that other customers are running across a pool of servers but they want their data to be isolated or *compartmentalized* from other customers' data for competitive, privacy, and integrity reasons.

Let's look at a variety of business problems that lend themselves to this form of compartmentalization. Whenever you replicate or move data to provide a new security container or isolation point, it's an opportunity to consider *labeling* the data and, in turn, saving on processor, network, storage, and administrative expenses. Suffice to say that the future is getting closer. The labeling of data and application identities provides the means for both the aggregation and compartmentalization of data.

One of the primary advantages of executing on zSeries software is your ability to host large databases on behalf of transaction programs or other application servers. Let's take a closer look at how you can compartmentalize data using your operating system, security server, and database technology, and in doing so, facilitate some database aggregation that reduces execution costs. It is the strength of the security on z/OS and within DB2 that provides this functionality. In particular, DB2® UDB for z/OS® Version 8 provides row-level security and z/OS V1.5 with Security Server (RACF) provide the operating system and security services that make this *database on demand* capability whole.

Government data compartmentalization

Intelligence communities within many government agencies have adopted the concept of compartmentalization of data to ensure that one compartment protects the need-to-know aspect of a particular piece of information. Management procedures and technology are utilized to protect both the reads and writes of this compartmentalized data. Many times, the term "labeled data" is used to reflect

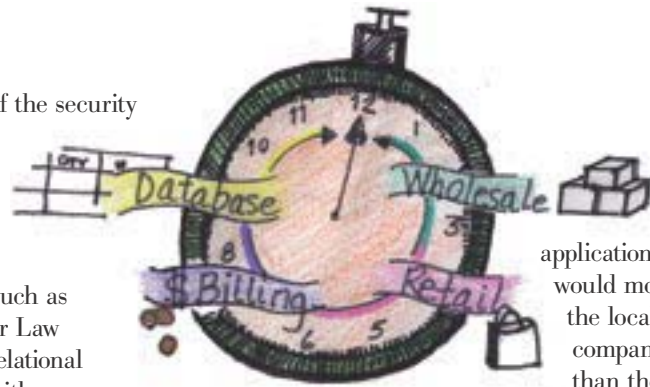
the combination of the security hierarchy (such as Secret, Internal, and Unclassified) and a particular category of data (such as Tax Department or Law Enforcement). A relational database is built with a collection of columns and rows. You can apply security labels to the columns and rows and provide a level of privacy or protection that isolates elements of the database from users running with differing security labels. It is the same form of labeling used by several government agencies that you can leverage to provide a commercial version of database compartmentalization that meets the needs of *on demand computing*.

Telephone billing on demand

Let's begin with a couple of examples. In the United States, a business or consumer has a choice of local and long distance telephone carriers. The local carrier is responsible for providing the connectivity to the long distance carrier. Each local carrier has customers with a wide variety of long distance carrier users. The local carrier captures information necessary for the long distance carrier to provide billing services to the consumer. When looking at the flow of data needed to provide long distance billing services, the data moves several times and consumes processing power with each move.

For example, the local provider must sort through all of its consumers by long distance carrier. Then the billing information for that long distance carrier must be aggregated and delivered to the long distance carrier. The long distance carrier receives the information, loads it into its own database, and then executes the billing application. In this flow, the data moves to the long distance billing application. Each month, a variable amount of data moves between the two telephone companies.

If the local telephone company provided an *on demand billing* service, the long distance companies' billing



application would move to the local phone company, rather than the billing information

moving. The long distance company would learn the schema of the database, the specific rows and columns necessary to process their billing records. (See Figure 1.) The local phone company would recognize the identity of the billing application as having arrived from a specific long distance carrier. That identity would be used to associate a label to that long distance carrier's billing application

Local phone companies billing records

Allows long line companies to run locally
Seclabel="longline_name"

DB2_SECURITY_LABEL_EXT	calling #	called#	duration
Long line 1			
Long line 2			
Long line 3			
Long line 1			
Long line 3			
Long line 4			
Long line 2			
Long line 3			
Long line 1			
Long line 4			
Long line 2			

Figure 1 - Example using security labels for billing on demand

at execution time. The database would only provide information to the billing application that was authorized to access it.

In this way, both the local and long distance carrier could save in billing processing. The local carrier would not consume servers and networks to move the data to the long distance carrier and in turn, it would be compensated for the processing done on behalf of the long distance carrier. The long distance carrier would reduce its server and network infrastructure while paying for the billing

application cycles on the local phone company's system at a fraction of the cost of hosting its own database. The data could then be available for other business analysis, again at a fraction of the cost. And throughout this database processing, the security labels associated with the long distance company applications would inhibit other long distance carriers from viewing their data.

Application serving on demand

Another example could be considered *wholesale on demand* computing. In this example, a services business acquires a collection of servers and hosts a specific application and its associated data on that server infrastructure. The services business then sells "subscriptions" for the application to other businesses or consumers. These application users, the "buyers", want their data isolated from other buyers but they choose to subscribe to the application service because they can not afford or don't have the skills to host their own version of the application's computing infrastructure.

By associating labels with each subscriber, the database can be consolidated across multiple buyers while retaining the appropriate levels of compartmentalization. (See Figure 2.) This allows the services business to save disk and management resources by consolidating the needs of multiple buyers into fewer databases. In addition, if a specific buyer asks for extensions to the database schema, this can be accomplished simply and easily—without migrating the data, taking down the application, or compromising the compartmentalization of the data. This can be accomplished with an online DB2 schema alteration and an online database reorg.

Financial services on demand

The final example addresses the large financial services corporation. Government regulations may inhibit one business unit from seeing personal, consumer information associated with another business unit. However, a subset of this information may be valuable for data mining, for example, identifying trends and developing new business services. By aggregating the data (see Figure 3) and labeling the columns of data, you can make a subset of the information more readily available without compromising the sensitive information that government

regulators are attempting to protect.

In this way, your financial services business can save the server and network costs associated with replicating data across business units, while still providing the compartmentalization that the government regulations are mandating. This could be considered a *retail on demand* solution.

Conclusion

Security labeling of data and application identities provides the means for both the aggregation and compartmentalization of data. There are a variety of other business problems that lend themselves to this form

of compartmentalization. Next time you need to replicate or move data to provide a new security container or isolation point, consider labeling and saving on processor, network, storage, and administrative expenses. Take a look at your database organization and flow of data between application servers. Maybe labeling your data can provide additional security and deployment savings to your business.

Outsourcer running an application practice
Common DB schema across customers
Seclabel="customer_name"

DB2_SECURITY_LABEL_EXT	COL1	COL2	COL3
Customer A			
Customer B			
Customer A			
Customer B			
Customer C			
Customer D			
Customer E			
Customer A			
Customer B			
Customer D			
Customer E			

Figure 2 - Using security labels for application serving on demand

Large company managing HR for subsidiaries
"corporate phone book"
Seclabel="subsidiary_name"

DB2_SECURITY_LABEL_EXT	COL1	COL2	COL3
Subsidiary 1			
Subsidiary 2			
Subsidiary 3			
Subsidiary 1			
Subsidiary 4			
Subsidiary 2			
Subsidiary 3			
Subsidiary 4			
Subsidiary 1			
Subsidiary 2			
Subsidiary 4			

Figure 3 - Using security labels for financial services on demand