



**SQLDBI Version 3.0**

**Technical White Paper**

## Contents

- **Overview: Defining the Business Environment**
  - What is a Line-of-Business Application?
  - Status Quo: How Line-of-Business Applications are Developed Today
  - Changing the Status Quo: SQLDBI
- **SQLDBI Technical Description**
  - Overview and Environment
  - Application Designer
  - User Solution
    - User Solution: Special User Interface Features
  - Virtual Private Database Engine
  - Supporting Components
    - Tracking and Logging
    - Document Management
    - Custom Controls and Utilities
    - Business Intelligence Integration
    - Notifications Engine
    - Calculations Engine
    - SQLDBI Publisher

# Chainbridge Technologies – SQLDBI 3.0

## What is a Line-of-Business (LOB) Application?

Large organizations, whether they are companies or government agencies, have several tiers of information technology needs:

- Physical Infrastructure (Fiber, Routers, Servers)
- Network Security and Access (Firewalls, Antivirus software, VPN, Shared Drives)
- Enterprise Applications (Directory Services, Accounting, CRM, Supply Chain)
- Line-of-Business Applications (SQLDBI)
- Desktop Productivity Applications (Word Processing, Spreadsheets, Printing)

Both the Enterprise Level and the Desktop Productivity applications are well defined and well served with vendors like SAP, IBM and Oracle at the Enterprise, while Microsoft dominates the desktop. The problem for many organizations lies in the gap between these two. Consider these examples:

- A federal government agency is responsible for responding to a WMD attack on US soil. It has several thousand personnel scattered around the country, but the people are attached to 30 different facilities managed by 50 different contractors with dozens of funding sources.
- A pharmaceutical company's sales managers and field representatives need to view and analyze their sales by clinic, chain, product and territory, then cross-reference the sales data against clinical efficacy data. The staff members need to collaborate and trade information.
- An engineering company has dozens of project engineers managing scores of field technicians performing sampling and testing. The company needs an end-to-end process that allows project managers to schedule and dispatch technicians, perform the sampling, ship the samples to a lab, record and analyze results, and report to the customers. Thousands of samples are managed and analyzed for a hundred different projects at a dozen different labs each week.

These examples are drawn from widely divergent industries and on the surface seem to have little in common. However, they do share a number of important similarities:

- They all revolve around a large amount of information that can be represented in a database.
- The type of information managed is highly specialized to both the organization as a whole and the departments and teams within it.
- The systems require both reporting and collaborative data entry.
- There is a diverse set of users with specific data access rules across many locations.
- The data managed is extremely sensitive and confidential, and it is crucial that it can neither be compromised by malicious individuals outside the system, nor inappropriately penetrated by authorized users of the system.
- The applications are mission-critical and users will not tolerate downtime or failures.
- The applications must be able to expand and change in response to modifications to the business process and organization.
- They are typically funded at a departmental level, so solutions must be cost-effective and deployed quickly.

## Chainbridge Technologies – SQLDBI 3.0

### Status Quo: How Line-of-Business Applications are developed today

Line-of-Business applications (LOBs) automate and control the day-to-day operations of an organization. Even if there isn't a management system currently in place, there will at least be established procedures that make some use of technology. This might entail paper forms that are scanned or faxed, spreadsheets stored on a shared drive, or a fairly comprehensive legacy system using Microsoft Access or Lotus Notes. If an organization has made a commitment to put an operations management system in place, there are a number of traditional methods they might employ. However, each has significant drawbacks:

- Traditional operations management - Paper Forms, Faxes and Emails
  - Surprisingly, this is the way many organizations still meet their departmental needs. They are easy to implement and have almost no infrastructure cost.
  - It is impossible to enforce standards and difficult to train staff. Since there is no reporting or data aggregation, this type of process will result in redundant and error-prone data entry.
- Legacy System (e.g., Microsoft Access, Lotus Notes, etc.)
  - This is the natural evolution from Forms, Faxes and Emails. This is usually a fairly easy way to start implementing process automation, and the skill set required to develop the system may be available internally.
  - However, this is the very definition of a stovepipe. These systems are not scalable, usage quickly outgrows capability, and they cannot integrate with other departmental or enterprise-level applications.
- Collaborative Portal (e.g., SharePoint, Software as a Service)
  - Portals and other Software as a Service applications are normally very easy and cost-effective to deploy an initial system.
  - These types of systems suffer from the same basic problem as COTS applications; specifically, the data structure is fixed and is extremely unlikely to meet the organization's business processes. They are very good at managing unstructured data like documents and links, but cannot manage highly custom, structured data in a format usable for analysis or reporting. Compounding the problem, while COTS applications can usually be modified, portal applications typically have a fixed or inaccessible data structure that cannot be easily read or expanded.
- COTS (Commercial-Off-The-Shelf)
  - From a management perspective, a COTS system may be very attractive. An initial system can normally be set up quickly, which allows the organization to leverage existing code and features.
  - For all but the most standardized processes like sales force automation or CRM (Customer Relationship Management), it is highly unlikely that an organization will find a COTS program that matches its unique Line-of-Business requirements. The solution is to either change the business to match the software, or customize the software to match the business process.

## Chainbridge Technologies – SQLDBI 3.0

- Business Intelligence Software (e.g., Cognos, Microstrategy, SAS, Business Objects<sup>1</sup>)
  - BI software can produce high-impact, impressive reports and analysis.
  - The primary problem with BI platforms is that they assume the information they are reporting or analyzing is already in place. Consequently, they do not address how the underlying data gets collected and managed, minimizing their utility in deploying operations-management systems.
- Enterprise Application Extensions (e.g., SAP, Oracle Financials, etc.)
  - This extension ensures that the application will be integrated into the rest of the Enterprise.
  - This approach either forces the Enterprise to conform to a department's needs, or forces a department to change the way it works to fit into the Enterprise. In addition, this method can be time- and cost-prohibitive.
- Custom Development – Internal or External Development
  - Similar to a COTS solution, engaging in a custom development effort may initially seem attractive. An external company may have expertise with similar processes, and internal developers will hypothetically already understand the problem. If successful, the organization will deploy a system that exactly meets its needs.
  - A custom development effort is potentially the most daunting option. To minimize risk, standard development methodologies emphasize a long and expensive requirements process. This can easily result in a system that is already operationally or technologically obsolete by the time it is delivered. Even if successful, the end product is complex custom code not easily maintained, meaning that the organization is dependent upon the original development team for rudimentary maintenance and expansion.

---

<sup>1</sup> Business Objects is now owned by SAP.

# Chainbridge Technologies – SQLDBI 3.0

## Changing the Status Quo: SQLDBI

SQLDBI (SQL Database Intelligence) is designed to address the problems associated with the traditional approaches to Line-of-Business process automation. The traditional approaches cannot meet the organization's particular needs (COTS, portals, Software as a Service), are not robust enough (legacy applications, Business Intelligence platforms), or are expensive and slow to deploy (customized COTS, custom development).

In contrast, SQLDBI is built around the following design principles:

- **Rapid prototyping and initial deployment:** Line-of-Business application requirements are notoriously difficult to adequately define. Instead of trying to fix this problem by employing an extensive requirements development process, SQLDBI allows a designer to quickly deploy a fully functional prototype in days or weeks. This allows clients to actually use a real system with familiar data and interfaces. This collaborative process is the best way to understand the organization's needs and quickly transform the prototype into a fully functional operations management system.
- **Reliability:** Because a SQLDBI-based system is built on a managed foundation currently trusted to run mission critical systems, senior managers can be confident that their operations management system will be secure, dependable, and will be well designed.
- **Long-term maintenance:** SQLDBI's core architecture is designed to be modified<sup>2</sup>, meaning that the underlying configuration structure automatically adapts itself in response to system modifications. This benefit will initially be realized during the prototyping and initial development stage because the system can be continually altered without introducing breaking changes. However, the real benefit will be realized post deployment because the application, built on a structured platform, can be continually modified and expanded without extensive redevelopment efforts.
- **Cost-Effective:** SQLDBI curtails the initial development stage and reduces the long-term maintenance burden, dramatically reducing professional service hours.

In this manner, SQLDBI allows an organization to deploy a fully functional, reliable, Line-of-Business application, which can be delivered much faster and maintained more easily than with any other traditional method.

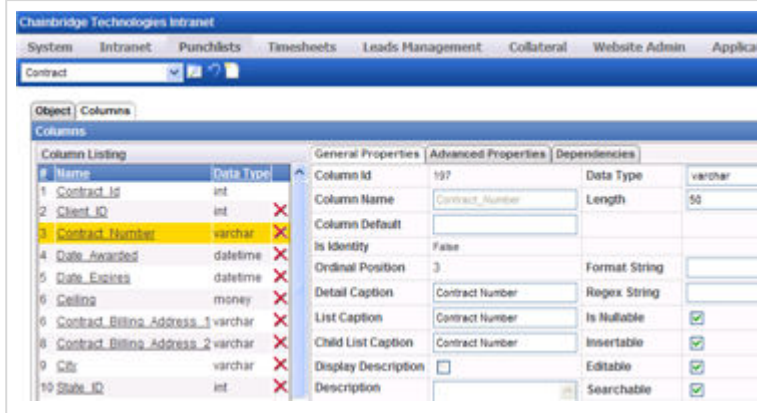
---

<sup>2</sup> SQLDBI's core architecture is separated into three components (the Application Designer, User Solution, and Virtual Private Database) that automatically adapt themselves in response to system modifications. These topics will be covered in depth later in this document.

# Chainbridge Technologies – SQLDBI 3.0

## What is SQLDBI – Overview and Environment

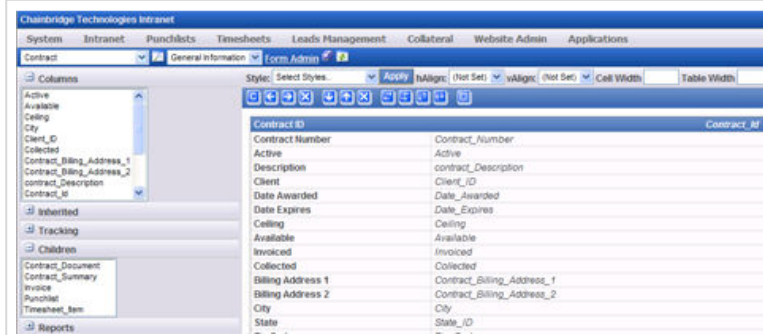
At the lowest level, SQLDBI is a server-side software product used to design, automate, and deploy Line-of-Business applications (LOBs). It runs within Microsoft's Internet Information Services (IIS) web server and is accessed through a standard web browser. All aspects of SQLDBI, both from the designer's and the end user's point of view, are accessed through familiar browser-based forms; there is no additional software required (e.g., ActiveX controls, client utilities, or browser plug-ins). Some typical interfaces used by designers and end users are provided below.



### Application Manager

*(Administrative interface)*

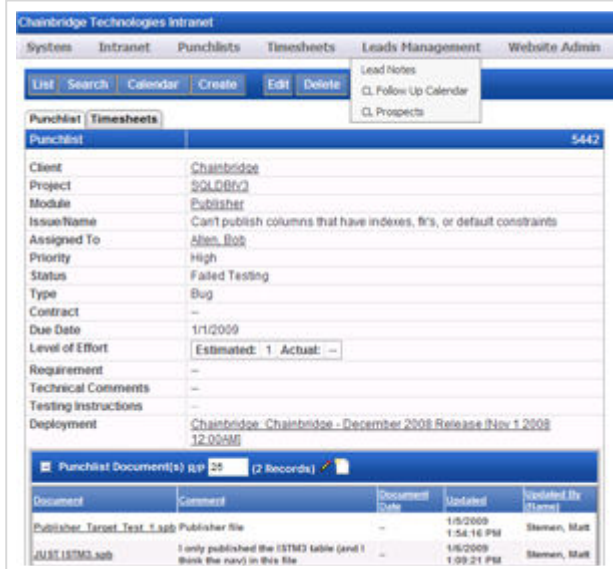
Utilized by a designer to create and modify database objects (tables, views and stored procedures) used by the application. This allows an authorized designer to remotely modify the database without having access to the server itself.



### Detail Form Builder

*(Administrative interface)*

Combined with the List Manager, this browser-based utility allows a designer to construct the forms used to access the application. With an easy to use, grid-based design surface, developers quickly place controls, grids, and tabs.



### Typical Detail Form

*(User Solution component)*

Once the designer has defined an Application Module and constructed some forms, the application is ready for use. Surprisingly, an end user can use the system while the designer is building it, seeing changes as they occur.

This form demonstrates some common SQLDBI detail form features including navigation buttons, cascading menus, tabbed forms, and editable child grids.

## Chainbridge Technologies – SQLDBI 3.0

On the server side, the SQLDBI installation is simply a matter of an XCOPY deployment into a configured Microsoft Internet Information Services (IIS) virtual directory<sup>3</sup> and connecting to an accessible SQL Server<sup>4</sup>. During normal operation SQLDBI does not require registry or disk write access<sup>5</sup>, and does not require any access outside of its own root virtual directory, resulting in a small server footprint.

By running entirely within IIS and requiring no additional client-side components, SQLDBI can easily coexist with all standard internet security technologies (e.g., SSL, client certificates, VPN, IP restrictions). In addition, because neither disk write access nor registry access is required, a SQLDBI application's network privileges can be tightly restricted. Finally, once a site is moved to fully deployed, the database login can be limited to a least privileged status with only read, write and execute access to a restricted set of tables, views, and stored procedures. In this manner, SQLDBI intrinsically employs a multilayered, security-in-depth approach to reduce the attack surface and minimize any impact resulting from a malicious attack.

### Security Note

SQLDBI based applications currently run in .com, .mil, and .gov environments, and manage both top-secret classified national security information as well as sensitive business confidential data.

SQLDBI applications have passed the NIST 800 security certification process, and are federally registered Systems of Record applications trusted to manage PII data for critical national security applications.

When a site hosting a SQLDBI instance is accessed, two components activate: the Database Configuration Manager (DCM) and associated Schema Discrepancy Repairer (SDR). Working together, the DCM and SDR interrogate the primary database and ensure that all required database components are correctly configured. This comprises a base set of ~100 tables and views, as well as ~50 stored procedures and functions. The DCM and SDR are also responsible for correctly modifying these components to support upgrades and service packs. As the site begins to build out, additional SQLDBI components are constructed as needed<sup>6</sup>. These components control all data access and entry, and are the foundation of the SQLDBI Virtual Private Database (VPD) system. With the exception of the database connection (stored in the web.connectionstrings.config), the entire application configuration is stored in the database itself.

The various configuration tables maintained by the DCM are edited using the various Application Designer tools. All configuration tables are well named and highly normalized. This allows easy access for documentation, bulk updates, or extension with custom components or external processes.

---

<sup>3</sup> SQLDBI relies upon Microsoft .NET version 2.0 or greater.

<sup>4</sup> All versions of SQL Server 2000, 2005, and 2008 are currently supported.

<sup>5</sup> SQLDBI includes document management capabilities. To eliminate the need to disk write access, these documents, are by default stored in the database. However, for systems with a large number of documents, this may be modified to write directly to disk or, if SQL Server 2008 is used, to disk via the FILESTREAM functionality. Either option will require elevated privileges for the process running the relevant application pool.

<sup>6</sup> Core SQLDBI components are prefixed with the characters "AB\_". Supporting components are suffixed with "\_AB". Supporting components include resource views, indexes, history and affiliation tables, and resource functions. All SQLDBI components are additions to the core database, and may be easily removed with a utility.

## Chainbridge Technologies – SQLDBI 3.0

The configuration information is used to construct a User Solution. When a user successfully signs into SQLDBI, the configuration is combined with the data-access rules defined in the Membership Manager<sup>7</sup> to initialize a new User Solution for the user. The User Solution then serves all user interface components (e.g., dashboards, navigation, lists, and forms), manages data access, and executes all database operations.

Consistent with security-in-depth practices, a given User Solution is only constructed with the components that the user has access to at the time that they log in. This is an important distinction because an authorized user is unable to circumvent the applications-defined security to access unprivileged data.

It is important to note that SQLDBI is not a code generator. The detail form example shown above<sup>8</sup> does not have a corresponding .html or .aspx file supporting it. The user interface (UI) for it is instead constructed at runtime using the information stored in the configuration tables<sup>9</sup>. While this does not have any measurable impact on performance, it has a significant security benefit because, depending upon the specific record being viewed, a given user may not even be aware that a particular section of the form or a set of fields could be available.

Consider a project management system accessible to external customers that has a developer comments tab, or a financial management section. A project manager likely may not even want most customers to be aware that such data is available and being managed, while a particular federal client demands that access. In this way, the rendered form is customized based upon the global rules, the user's privileges, and the specific condition of the record being viewed.

### Installation Environment

#### Web Server

- Windows Server OS
- .NET Framework 2.0+
- IIS Virtual Directory
  - VPN, SSL, IP restrictions, client certificates etc.
  - SQLDBI Site files
  - Config file pointed at an available SQL Server

#### Database Server

- Windows Server OS
- May run on the Web Server, or as a separate physical machine
- SQLServer 2000, 2005, or 2008

<sup>7</sup> The Membership controls the configuration of SQLDBI's Virtual Private Database.

<sup>8</sup> Please reference the Overview and Environment section.

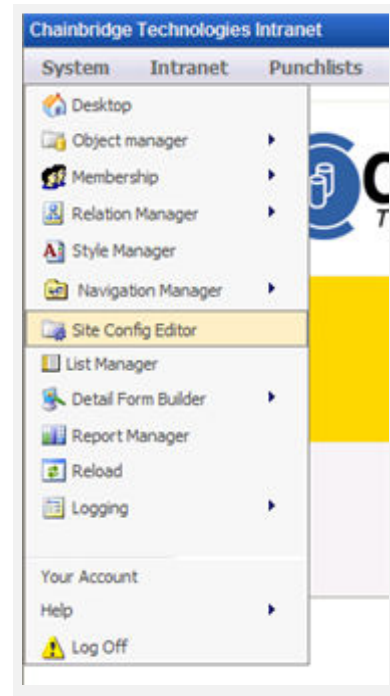
<sup>9</sup> For instance, detail form base configuration is stored in AB\_Detail\_Form, AB\_Detail\_Form\_Row and AB\_Detail\_Form\_Cell.

# Chainbridge Technologies – SQLDBI 3.0

## Application Designer

SQLDBI consists of three main components<sup>10</sup>: the Application Designer that edits the configuration, the User Solution that renders user interfaces, and the Virtual Private Database that controls data access.

The Application Designer is a browser-based development environment. At the lowest level, it reads and modifies the contents of the various configuration tables. When an authorized site administrator signs in, additional menu items become available under the System menu. From here, the designer can access all the Application Designer tools. In this way, the design environment is really a component of the actual deployed application<sup>11</sup>. This allows a designer to make modifications to a production or development site while clients or other developers are using the application, allowing them to collaboratively review and comment.



Key components of the Application Designer include:

- **Object Manager:** Creates and edits Application Objects, which are data sources like tables, views and stored procedures that the User Solution consumes when rendering user interfaces and managing CRUD<sup>12</sup>. Additional features include setting specialized field properties, display captions, formatting rules, and setting global access restrictions.
- **Workflow Editor:** Allows the designer to implement rules governing when particular records and fields are available, depending upon the state of the record and the identity of the current user. For instance, a rule could be developed stating that an Accounting Manager is only allowed to modify an invoice once its status has been set to Issued or Paid.
- **Membership Manager:** Administers users, groups, privileges and policies<sup>13</sup>. This is the main interface for configuring the Virtual Private Database engine described in detail later in this document.
- **Relation Manager:** Is similar to a referential integrity editor, but has many additional features that are used to configure how a given relation behaves. For instance, if a Contract is related to a Client, the Relation Manager would be used to configure how the Client lookup control is

<sup>10</sup> These are augmented by the various 'engines' that provide specialized functionality: the Publisher, Workflow Engine, Notifications Engine, etc. These will be discussed later in this document.

<sup>11</sup> The Application Designer requires dbo access for certain modifications like creating or modifying tables. Once the site is in production, the connection can be limited to read/write/execute and the relevant designer tools disabled to reduce the attack surface against the database.

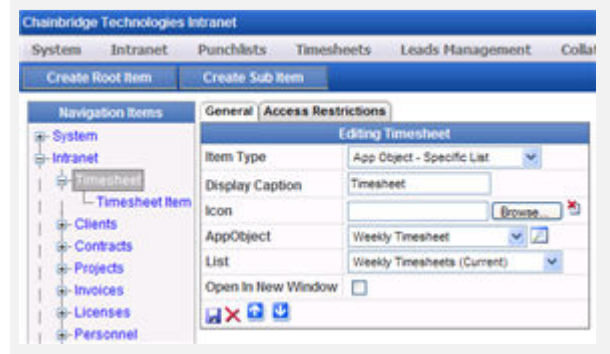
<sup>12</sup> CRUD refers to the basic UI functions of a database; Create, Read, Uppdate, and Delete.

<sup>13</sup> Please see the Virtual Private Database section later in this document

## Chainbridge Technologies – SQLDBI 3.0

displayed on the Contract form (e.g., Name [Street, City, Phone#]), which clients are displayed and when, and how the Contracts behave when placed on a Client form.

- **Style Manager:** Is a CSS (Cascading Style Sheet) editor that enforces consistency and controls the overall look of the application.
- **Navigation Manager:** Controls the top menu and the basic desktop display. Navigation items may be established with an unlimited complexity and can link to forms, lists, searches, calendars, custom utilities, or external resources. A given user will only be presented with a navigation item if they have access to the relevant target. In this manner, different types of users may be presented with very different navigation options.
- **Site Config Editor:** Is the main interface for various global settings for the overall application, ranging from the application's site title and welcome message to authentication and licensing settings.
- **List Manager and Detail Form Builder:** These two utilities control how the User Solution renders user interfaces. The List Manager controls field displays, sorts, filters and accesses restrictions for searches, lists, and child grids. The Detail Form Builder is a grid-based html editor that controls how a single record and all of its associated data (e.g., child records, reports, and custom controls) are displayed and managed.
- **Report Manager:** Allows external business intelligence components like Crystal Reports and SQL Server Reports to be bound to a SQLDBI application. This allows the application to seamlessly publish high-quality PDF reports as well as display graphs and charts.



# Chainbridge Technologies – SQLDBI 3.0

## User Solution

When a user successfully signs in, a server component in the application memory space named the AppManager instantiates a new User Solution for the authenticated user and stores it in their session memory space<sup>14</sup>. All requests from the user's browser are then routed through the User Solution, which is responsible for rendering all UI (menus, dashboards, reports, lists and forms) and managing all CRUD operations.

Client requests to the User Solution occur both through standard HTTP GET and POST requests as well as through AJAX (i.e., JSON, SOAP) calls to the server.

From the end user's perspective, the User Solution is the application; they will typically not even be aware that the Application Designer tools exist. The User Solution is presented as a familiar browser-based application. The UI is intentionally extremely simple and consistent. This has the benefit of greatly reducing the training and support burden. Once a user is familiar with any one of the modules, they will find that all subsequent modules work in the same manner.

### Utilitarian by Design

Typical SQLDBI usage is data-entry and search heavy; most users are clerical staff, managers, and their client counterparts. Because SQLDBI is used to build LOB operations management systems (as opposed to 'portal' or 'informational' web sites) the user interface is designed to allow users to find what they are looking for and get their work done as efficiently as possible.

A SQLDBI application is organized around Application Modules. An Application Module is a data source like a table, view, or stored procedure. Most of the user's experience with a SQLDBI site will revolve around working with UI components bound to a particular Module. These fall into 3 categories<sup>15</sup>:

**Lists and Child Grids:** Lists are familiar tabular data grids. A child grid is simply a list that appears on a detail form, e.g., invoice items appearing on an invoice. All lists have automatic pagination and sorting, can be exported to Excel, and quickly navigate to other associated lists. For instance, a Punch List Module might have a series of configured lists for Open, Completed and Overdue Punch List items.

**Detail Forms:** A detail form is used to view or edit a single database record as well as insert new records. Typically, a user will click on a key or navigation link in a list or child grid to open a given detail form. A detail form is usually displayed as a grid with 2 or more columns and as many rows as are needed; an image of a typical detail form can be viewed in the Overview and Environment section above. Individual cells can contain a variety of controls, the most common of which are text and images entered by the designer, as well as fields (both captions and data) from the underlying database table. Also common are child lists displaying associated data (e.g., invoice items on an invoice), and report components bound to a business intelligence provider like SQL Server Reporting Services or Crystal Reports. Detail forms can have multiple tabs, which can be hidden using standard access restrictions.

---

<sup>14</sup> This does not require client-side cookies and is invisible to the user's browser.

<sup>15</sup> SQLDBI also supports calendar functionality, but this in practice is simply an alternate method to filter a list.

## Chainbridge Technologies – SQLDBI 3.0

Searches: Every Application Module automatically has at least one search utility, although it can ultimately have an unlimited number. The available searches will be displayed in a drop down at the top of the search utility, as well as at the top of appropriate lists and child grids.

The search utility allows the user to select which fields they would like to have displayed, the order those fields are displayed (from left to right), how the data is sorted, grouping or summaries to be applied, and which filtering conditions to use. The filtering conditions are extremely flexible. For instance, in the example to the right, the user is requesting to see all the Punch List items which:

1. Are assigned to either Bob Allen or Mark Hildreth
2. AND have a due date between 1/1/2008 and 3/1/2008 OR 5 days plus or minus from today
3. AND have descriptions containing the word "Publisher" OR the word "Deployment"

If configured, the user can also easily search on child, grandchild, or inherited data. For instance, the example above could be expanded to include:

4. AND with a Client whose address is in PA, MD, or VA
5. AND has a Punch List Comment entered in the past 30 days.

Once the user presses the "Process Search" button, they will be taken to a list form showing the data matching their request. In the drop down at the top of the form will be a new entry in the format "Search Name (1)". Every time the user searches, their search conditions are saved in the current session. This allows the user to toggle between searches and further refine the filters to find exactly the information they are looking for. Finally, the user can save one of their searches permanently, giving it a name and description that is available only to them. This allows a user to customize their application to meet their particular needs.

Also, since all SQLDBI search modules function identically, once a user is familiar with one Application Module search utility, they will be familiar with all of them. As with other aspects of SQLDBI, this greatly reduces training and support burden.

### Self Service Reporting

By adjusting the search filters, the user has extremely fine control of which data is displayed, but without ever making a call to a database administrator.

By using the various export tools, users can even generate and save their own reports without the assistance of a business intelligence developer.

# Chainbridge Technologies – SQLDBI 3.0

## User Solution: Special User Interface Features

While SQLDBI's end user interface will be extremely familiar to anyone accustomed to working with browser-based forms, there are a number of specialized features that have been developed over time to enhance the user's experience and improve data entry. Some examples include:

- **Quick Edit:** By holding down the control key while clicking on a piece of data the user is viewing, SQLDBI swaps in an editable control (e.g., text box, checkbox, drop down, etc.). When the user edits the new control, their changes are immediately saved and sent to the server. Quick Edit works everywhere; details forms, lists, child grids, calendars, etc. This is especially helpful for clerical staff because they can, for instance, do a search of all of the outstanding invoice issues in a particular date range and quickly set their status to Paid without having to navigate to every single record.
- **State Management:** One of the main criticisms many users have with web-based applications is that they are stateless; e.g., if they are typing into a form and inadvertently leave the form, all of their changes are lost. SQLDBI maintains a server-side State Manager that keeps track of all of the user's data entry. If they open a form for edit or insert, they can enter information and then navigate to another form. When they return to the original record, they will find that it is still open for editing and all of their entered information is waiting for them.
- **Advanced Validation:** Since the State Manager is synchronized with the user's form, it is possible to implement much more advanced formatting and validation code on the server than could be handled through simple client-side validation.
- **Inline Edit and Insert:** All child grids support multiline insert, edit and delete directly from the parent detail form without having to navigate to a child record and then navigate back to the parent record. This is particularly useful when combined with Quick Edit. Consider a Punch List system that has Tickets assigned to individual users. With inline editing, the PM can quickly find a given Punch List and reassign all of one person's tickets to another if a staffing change occurs.
- **Dependent Refresh:** One of the most common, and problematic, features in a web application is a dependent dropdown selection; e.g., selecting a country filters a narrowed list of states/provinces, further filtering a list of counties/towns. Dependent Refresh allows a designer to easily implement what would normally be very complicated pieces of UI.
- **Custom Controls:** Even with all of SQLDBI's features, the application may require functionality not provided out of the box. To handle these situations, SQLDBI provides a well-defined API and control tree to allow a .NET developer to author and insert their own custom controls (.ascx's) directly into detail forms and lists. A number of examples are provided out of the box that can be used as templates, including the Affiliation Viewer and History Viewer.

### **Fat Client experience, Thin Client environment**

SQLDBI-based applications are most commonly used for internal, process automation tasks like operations management and human resources. Consequently, even though they are using a browser, end users expect all of the features they would have in a traditional fat client application installed on their desktop. To deliver that level of seamless functionality, SQLDBI makes heavy use of AJAX and SOAP WebServices.

# Chainbridge Technologies – SQLDBI 3.0

## Virtual Private Database Engine

The Application Designer configures the behavior and appearance of a SQLDBI Application, and the User Solution renders all UI and mediates CRUD, but the most powerful component of SQLDBI is the Virtual Private Database Engine, or VPD.

The VPD was constructed in response to extremely common and serious problems:

- It is crucial to control data access in a LOB system accessed by multiple users.
- Access rules are complicated and are almost never well defined when application design starts.
- As the application evolves to solve more problems for more types of users, so do the rules.
- Layering access restrictions onto a system near completion, or already in production, is at the very least complicated and expensive and, at the worst, is error prone and can easily result in sensitive data leaking to unauthorized users.

To solve these problems, SQLDBI separates the component that designs the application (Application Designer) from the component that renders the application (User Solution), from the component that reads and writes data (VPD). This allows the application's design to proceed, with the data-access security being added or modified at any time during the development process, or even after the application is already in production.

Consider a human resources system with these requirements:

- All users can view their own salary.
- A manager can view the salaries of his direct reports.
- Accounting staff can view the salaries of all employees in their departments.
- Accounting managers can edit the salaries of all employees in their departments.
- Accounting executives can edit the salaries of all employees.

Personnel Salary Data		
Last Name	First Name	Salary
Allen	Chris	\$85,000
Hildebrand	Mark	\$54,000
Cavanaugh	Paul	\$35,000
Palmer	Chris	\$80,000
Lindsey	Chris	----
----	----	----

Think of the VPD as a censor that, upon receiving a request for data from the User Solution, retrieves the records the user is allowed to see, and annotates the data set to indicate which particular access (e.g., editing) the current user has on each record as well as on every field in that record. The User Solution does not care how those rules are defined; all it cares about is the marked up data set that the VPD returns. Consequently, the rules can be changed at any time with no impact upon the system; the various user interfaces and menu items simply adapt the next time they are accessed.

The main interfaces used to configure the behavior of the VPD are the Object Manager, the Membership Manager, the Relation Manager, and the Affiliation Manager. Taken together, these interfaces author configuration rules stored in the configuration tables. When SQLDBI starts, the DCM<sup>16</sup> interrogates the configuration and constructs all required database access rules.

<sup>16</sup> Database Configuration Manager; discussed in Overview and Environment  
Chainbridge Technologies – 8229 Boone Boulevard, Suite 100, Vienna VA 22182 - www.ChainbridgeTech.com  
Page 15 of 18

# Chainbridge Technologies – SQLDBI 3.0

## Affiliations, Membership Binding, Policies, and Privileges

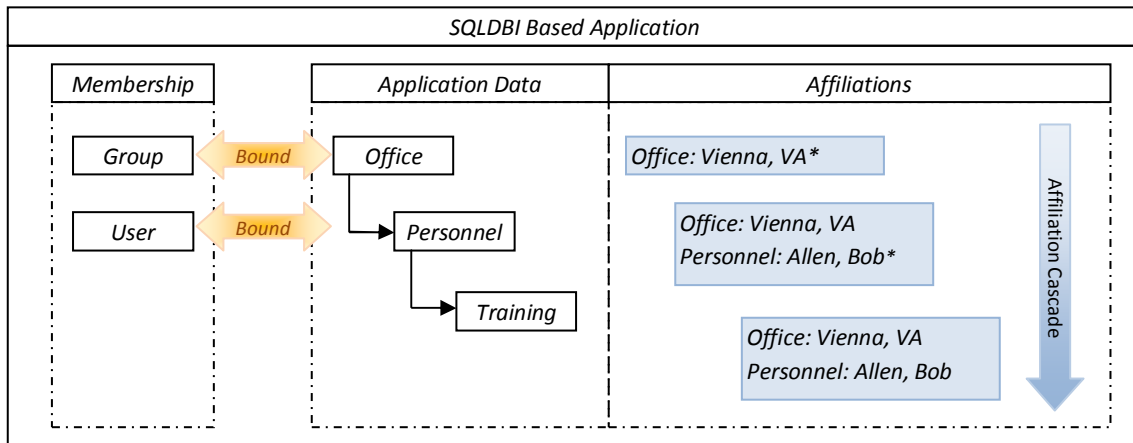
The core innovation powering the VPD is the Affiliation System. An affiliation is analogous to a Windows-based Access Control Level (ACL) applied contextually to a relational database system. Windows ACL's are built on users, groups, drives, folders and files, with users and groups defined in the directory system and drives, folders and files managed by the file system. A user or group 'token' can be attached to a particular resource like a folder, which can then 'cascade' to contained elements like folders and files. The group can then acquire additional members (users or other groups), which cause the relevant tokens to automatically cascade to the appropriate resources.

SQLDBI maintains a user and group system similar to Windows, which is managed by the Membership Manager. Users can be placed into multiple groups, which can in turn be nested in additional groups.

Crucially, SQLDBI also allows tables to be 'bound' to the membership system. Specifically, this means that every record in a 'bound' table will have a corresponding record maintained in the membership system. In that manner, a given membership token (i.e., a user or group) can be affiliated with a specific database record, essentially attaching an ACL to an individual database record.

Using the Relation and Affiliation Managers, these ACL's can then be directed to cascade to descendent data. Consider the following example:

**User Authentication Control**  
SQLDBI's membership system can also be linked to an existing Active Directory (or other directory server), allowing authentication and group membership to be centrally controlled. Consequently, SQLDBI can operate in a mixed authentication mode with internal users controlled by Active Directory and external users controlled by SQLDBI itself.



In this case, both the Office and Personnel tables are membership bound to, respectively, groups and users. This results in a corresponding Office and Personnel affiliation being bound to the respective database record, as indicated by the " \* ". Since Personnel is a child of Office, the Office affiliation can be configured to cascade to the dependent Personnel record. Similarly, both the Office and Personnel affiliations can be configured to cascade to the dependent Training records. In this way, each Training is affiliated with a Personnel and Office membership token.

## Chainbridge Technologies – SQLDBI 3.0

Once the affiliation model is defined, the designer can create a policy. A policy defines a set of privileges held by the users who are members of the group. Since users can be members of multiple groups, a series of cascading policies may be defined<sup>17</sup>. For instance, in a very simple example the designer could create a group named General Staff and construct a policy such that members may:

1. View all offices
2. View all personnel in their assigned offices
3. Edit their own personnel record
4. View, edit and delete their own personal training records

Once a policy has been authored, it is applied to all applicable users as privileges. A user's individual privileges may be granted directly, inherited from one or more policies, or a combination of the two. Modifying a policy immediately cascades to all relevant users and groups.

Finally, relations, affiliation inheritance rules, membership binding, policies, and privileges may be modified at any time during or post development without breaking the application. This allows an extremely powerful, comprehensive and flexible membership and privileges system to be developed for an application and modified as needed without a redevelopment effort and without requiring any changes to the other portion of the SQLDBI Application. The next time the application is accessed, SQLDBI will interrogate the VPD configuration and will automatically re-examine and modify or construct all required components.

### **Additional Access Security Mechanisms**

In addition to the basic data access functionality provided by the VPD, SQLDBI provides an access restriction model that applies to several aspects of the system. This allows global restrictions to be put in place, tied either to groups or individual users. Items that can be globally restricted include tables, fields, navigation and desktop items, reports, lists, and detail form tabs.

Finally, for advanced systems, custom affiliations and workflow may be integrated into the base SQLDBI VPD framework.

---

<sup>17</sup> Privileges awarded from multiple policies are additive.

# Chainbridge Technologies – SQLDBI 3.0

## Supporting Components and Capabilities

Beyond the base Application Designer, User Solution, and Virtual Private Database, SQLDBI includes a number of additional components and capabilities that enhance a completed solution. Some include:

- Tracking and Logging Features
  - Extensive user data entry tracking, both at the record and field level
  - Report and utility usage history
  - User login history
  - Error history
- Document Management: Document upload capability may be added to any table simply through the inclusion of an image or binary SQL Server data type. SQLDBI interprets these fields as document upload fields and renders the appropriate UI for accessing them. Documents may either be stored in the database or, alternately, in the file system.
- Custom Controls: Detail forms, lists, and dashboards may incorporate custom .NET controls to extend the base functionality.
- Business Intelligence Integration: SQLDBI can be bound to both Crystal Reports and SQL Server Reporting Services (SSRS) to generate high-quality PDF reports. In addition, SQL Server Analysis Services (SSAS) products consumed by SSRS can be displayed on detail forms and dashboards to add high impact, data-driven graphics and summaries.
- Notifications Engine: Integrated with SQL Server DBMail and SQL Server Integration Services (SSIS), the Notifications Engine allows a SQLDBI-based application to issue notifications to authorized users (email, text, pager, fax, etc.) in response to events (i.e., triggers), on a regular schedule or contextually based upon certain conditions being met.
- Calculations Engine: Automates many of the general database developer tasks normally handled in views or triggers like rollup calculations, data entry constraints, and custom lookup rules.
- SQLDBI Publisher: Allows a SQLDBI configuration (whole or partial) to be moved between SQLDBI applications. This is crucial when publishing changes from a development or test environment to production. However, it is also useful in setting up blank applications based upon previously created SQLDBI templates, essentially converting an existing SQLDBI application into a COTS product. Multiple components can be published to a single site, incrementally adding, for instance, personnel management, training, resumes, and performance review components.